

GUO JIA SHI FAN XING GAO ZHI YUAN XIAO JIAN SHE XIANG MU CHENG GUO

国家示范性高职院校建设项目成果

计算机专业系列

.NET Web 应用开发

郭永洪 叶青松 裴 拯 编著

眭碧霞 主审

清华大学出版社

国家示范性高职院校建设项目成果·计算机专业系列

.NET Web应用开发

郭永洪　叶青松　裴　拯　编著
眭碧霞　主审

清华大学出版社
北　京

内 容 简 介

本书以开发"网上书店"这一项目为案例,介绍.NET Web 应用开发技术。

全书共分 10 个具有递进关系的子项目。项目 1 介绍简单 Web 网站设计。项目 2 介绍风格统一的 Web 网站设计。项目 3 介绍与用户交互性网站设计。项目 4 和项目 5 通过图书查询和管理功能的实现,介绍 ASP.NET 2.0 数据显示和访问技术。项目 6～项目 8 通过网上书店读者注册管理、购物车管理和订单管理业务流程的实现,介绍 ASP.NET 2.0 注册登录控件的使用、个性化配置和分层软件架构技术。项目 9 通过网上书店 Web 报表设计介绍水晶报表技术。项目 10 使用 ASP.NET 2.0 Web 认证和授权技术整合网上书店各个功能模块。ASP.NET 2.0 知识贯穿在项目设计和分析过程中。

本书既可作为高职高专院校计算机软件技术专业的教材,也可以作为计算机软件开发人员的参考用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。
版权所有,侵权必究。侵权举报电话:010-62782989　13701121933

图书在版编目(CIP)数据

.NET Web 应用开发/郭永洪,叶青松,裴拯编著.--北京:清华大学出版社,2011.2
(国家示范性高职院校建设项目成果.计算机专业系列)
ISBN 978-7-302-24554-4

Ⅰ.①N…　Ⅱ.①郭… ②叶… ③裴…　Ⅲ.①计算机网络－程序设计－高等学校:技术学校－教材　Ⅳ.①TP393

中国版本图书馆 CIP 数据核字(2011)第 009267 号

责任编辑: 田　梅
责任校对: 李　梅
责任印制:

出版发行: 清华大学出版社　　**地　址:** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编:** 100084
社　总　机: 010-62770175　　**邮　购:** 010-62786544
投稿与读者服务: 010-62776969,c-service@tup.tsinghua.edu.cn
质　量　反　馈: 010-62772015,zhiliang@tup.tsinghua.edu.cn
印 刷 者:
装 订 者:
经　销: 全国新华书店
开　本: 185×260　　**印　张:** 16.5　　**字　数:** 396 千字
版　次: 2011 年 2 月第 1 版　　**印　次:** 2011 年 2 月第 1 次印刷
印　数: 1～　000
定　价: 　.00 元

产品编号:033903-01

编 委 会 成 员

主　任：邓志良

副主任：闵　敏　束传政

成　员：赵佩华　曹建庆　杨　诚　陈剑鹤
薛苏云　眭碧霞　陈必群　秦益霖
赖华清　周　敏　宋　卫　庞　杰

秘　书：赵佩华　田　梅

出版说明

特色教材建设是推动课程改革和专业建设的基础，是提升人才培养质量的重要举措，也是高职院校内涵建设的重点之一。

2007 年，经教育部、财政部批准，常州信息职业技术学院进入 100 所国家示范性高职院校建设行列。开展示范院校建设以来，学院坚持以科学发展观为指导，针对市场设专业，针对企业定课程，针对岗位练技能，围绕区域经济建设、信息产业发展的实际需求，全面推进以“三依托、三合一”为核心的工学结合人才培养模式改革，强化职业素质和职业技能的培养，构建了具有学院自身特色的校企合作管理平台，在培养高素质技能型人才、为服务区域经济等方面取得了显著成效。

为展示课程建设成果，学院和清华大学出版社合作出版了常州信息职业技术学院特色教材 30 部，这也是学院示范院校建设的成果之一。作为一种探索，这套教材在许多方面还不尽成熟和完善，但它从一个侧面反映了学院广大教师多年来对有中国特色高职教育教学，特别是教材建设层面的创新与实践，希望能对深化以职业能力培养为核心的专业改革、切实提高教育教学质量发挥应有的作用。

在人才培养模式的创新、课程改革和教材建设中，我们始终得到教育部、财政部、江苏省教育厅、财政厅和国家示范性高职院校建设工作协作委员会等各级领导、专家的关心和指导，得到众多行业企业、兄弟院校和清华大学出版社的大力支持，在此一并致谢！

常州信息职业技术学院

清华大学出版社

2009.6

FOREWORD

前　言

信息技术的核心是软件，软件开发需要的是软件人才，从事编码和测试基础性软件开发的程序员是社会紧缺的技能型软件人才。高职高专软件技术专业肩负着培养技能型、实用型、基础型软件人才的任务。为了满足社会对软件人才的需求，同时使高职高专软件技术专业毕业的学生基本符合软件企业工作的编码要求，我们编写了《.NET Web应用开发》一书。本书结合工作过程思想，以项目为导向、任务驱动为基础的教材组织方式，通过一个真实项目的开发过程，重点培养学生的编码实践能力。

教材选取"网上书店"的设计与开发过程作为贯穿教材的项目，将整个项目分解为10个具有递进关系的子项目进行讲解。每个子项目的讲解分为项目介绍、项目分析、相关知识、项目实施、项目总结、项目实训6个完整的环节。具体讲解项目实施时，依据子项目的内容分解为若干子任务，子任务的介绍按任务介绍、任务分析、任务实施、任务总结、课堂训练与拓展等环节介绍。知识点贯穿在项目和任务的介绍中，学生在完成项目的过程中学习知识。

本书编写过程中得到江苏富深协通数码技术有限公司、常州腾信数码科技有限公司的高级工程师的帮助，他们对于项目的选择、项目实训、任务设计提出了很多宝贵意见，在此表示衷心的感谢。

本书项目1～项目3由裴拯编写，项目4～项目6由叶青松编写，项目7～项目10由郭永洪编写，眭碧霞教授审阅了全书。由于笔者水平有限，错误和不妥之处在所难免，敬请读者不吝赐教。

编　者

2010年9月

CONTENTS

目录

项目1

创建简单的网上书店Web网站

1.1 项目介绍

第一个项目主要学习 ASP.NET 的基础知识，内容包括 ASP.NET 技术的发展背景、ASP 与 ASP.NET 的区别与联系、ASP.NET 2.0 的技术特点、C# 与 ASP.NET、.NET 框架等，并创建一个简单的网上书店 Web 网站。

1.2 项目分析

学习 ASP.NET 技术需要先对它的 IDE 集成开发环境 Visual Studio 有一个详细的了解。Visual Studio 2005 的 IDE 集成开发环境增加了许多功能协助创建网站。本项目首先介绍如何在 Visual Studio 2005 中新建网站、打开网站，然后介绍 Visual Studio 2005 集成开发环境中常用的几种窗口，接着按照编辑网页的顺序，依次介绍新建网页、编辑网页、运行网页等相关知识。通过本项目的学习，读者可以对 Visual Studio 2005 有一个比较清楚的认识，为以后的学习打下基础。

1.3 相关知识

1. 微软.NET 的历史

随着网络经济的到来，微软公司希望帮助用户能够在任何时候、任何地方、利用任何工具都可以获得网络上的信息，并享受网络通信所带来的快乐。.NET 战略就是为着实现这样的目标而设立的。

Microsoft .NET 平台的基本思想是将侧重点从连接到互联网的单一网站或设备上，转移到计算机、设备和服务群组上，使其通力合作，提供更广泛更丰富的解决方案。用户能够控制信息的传送方式、时间和内容。计算机、设备和服务能够相辅相成，从而能提供丰富的服务，而不是像“孤岛”那样，由用户提供唯一的集成。企业可以提供一种方式，允许用户将他们的产品和服务无缝地嵌入自己的电子构架中。

下面简单介绍一下.NET 技术发展的历程。

2000 年 6 月，微软总裁比尔·盖茨先生在一次名为“论坛 2000”的会议上发表演讲，描

绘了.NET技术的宏伟蓝图。

2002年1月，微软公司发布.NET Framework 1.0正式版。与此同时，Visual Studio .NET 2002也同步发行。

2003年4月23日，微软公司推出.NET Framework 1.1和Visual Studio.NET 2003。这些产品都是针对.NET 1.0的升级版本。

2004年6月，微软发布.NET Framework 2.0 Beta1和Visual Studio 2005 Beta1。同时，还发布多个精简版(Express Edition)，其中包括Visual Web Develop 2005、Visual Basic 2005、Visual C# 2005和SQL Server 2005 Express Edition等。

2005年4月，微软公司发布Visual Studio 2005 Beta2测试版。

2005年11月，微软公司发布Visual Studio 2005和SQL Server 2005正式版。

2008年2月，微软公司正式发布产品Visual Studio 2008。

2. ASP.NET的历史

在讲述ASP.NET历史之前，先回顾一下ASP。ASP的第一个版本是0.9测试版。它给Web开发带来一阵风暴，它能够将代码直接嵌入HTML，使得设计Web页面变得更简单，并且通过内置的组件能够实现强大的功能，最明显的就是ActiveX Data Objects(ADO)，它使得建立一个动态页面非常简单。

接下来是Active Server Page 1.0，它作为IIS的附属产品免费发送，并且不久就在Windows平台上广泛使用。ASP与ADO的结合使用使开发者很容易地在一个数据库中建立和打开一个记录集，这无疑是它很快就被大众接受的原因。

1998年，微软公司又发布了ASP 2.0。ASP 1.0和ASP 2.0的主要区别是外部的组件需要实例化。有了ASP 2.0和IIS 4.0，我们就有可能建立ASP应用了，而且每个组件就有了自己单独的内存空间。内置的Microsoft Transaction Server(MTS)也使得制作组件变得很简单。

微软公司接着开发了Windows 2000操作系统，这个Windows版本带来了IIS 5.0以及ASP 3.0。此次并不是简单地对ASP进行补充，核心的不同实际上是把很多的事情交给COM(Component Object Model)来做。在Windows 2000中，微软结合了MTS(Microsoft Transaction Server)与COM核心环境做出了COM+，这就让主机有了一种新的方法来使用组件，同时也给主机带来了更强的稳定性，成为一个可以升级的且效率高的工作平台。IIS 5.0在表面上似乎没有更改什么，但是在接口方面改动比较大。在内部，它使用COM+组件服务来对组件提供一个更好的执行环境。

有了这些，微软公司又推出了ASP.NET，它不是ASP的简单升级，而是Microsoft推出的新一代Active Server Pages。

目前，微软发布的.NET最新版本是3.5。.NET 3.5的发布是.NET技术走向成熟的标志。尤其是用于Web应用程序开发的核心技术，ASP.NET 3.5更是万众瞩目，不断吸引着越来越多的目光。

3. Web页面简介

使用ASP.NET网页可以为网站创建动态内容。通过使用静态HTML页(.htm或

.html 文件),服务器读取文件并将该文件按原样发送到浏览器,以此来满足 Web 请求。相比之下,当用户请求 ASP.NET 网页(.aspx 文件)时,该页则作为程序在 Web 服务器上运行。该页运行时,可以执行网站要求的任何任务,包括计算值、读写数据库信息或者调用其他程序。该页动态地生成标记(HTML 或另一种标记语言中的元素),并将该标记作为动态输出发送到浏览器。

4. 回发和往返行程

ASP.NET 页面作为代码在服务器上运行。因此,要得到处理,页面必须配置为当用户单击按钮(或者当用户选中复选框或与页面中的其他控件交互)时提交到服务器。每次页面都会提交回自身,以便它可以再次运行其服务器代码,然后向用户呈现其自身的新版本。

ASP.NET 网页的处理循环如下。

(1) 用户请求页面(使用 HTTP GET 方法请求页面)。页面第一次运行,执行初步处理(如果已通过编程让它执行初步处理)。

(2) 页面将标记动态呈现到浏览器,用户看到的网页类似于其他任何网页。

(3) 用户输入信息或从可用选项中进行选择,然后单击按钮(如果用户单击链接而不是按钮,页面可能仅仅定位到另一页,而第一页不会被进一步处理)。

(4) 页面发送到 Web 服务器(浏览器执行 HTTP POST 方法,该方法在 ASP.NET 中称为"回发")。更明确地说,页面发送回其自身。例如,如果用户正在使用 Default.aspx 页面,则单击该页上的某个按钮可以将该页发送回服务器,发送的目标则是 Default.aspx。

(5) 在 Web 服务器上,该页再次运行,并且可在页上使用用户输入或选择的信息。

(6) 页面执行通过编程所要执行的操作。

(7) 页面将其自身呈现回浏览器。

只要用户在该页面中工作,此循环就会继续。用户每次单击按钮时,页面中的信息会发送到 Web 服务器,然后该页面再次运行。每个循环称为一次"往返行程"。由于页面处理发生在 Web 服务器上,因此页面可以执行的每个操作都需要一次到服务器的往返行程。

5. 页面生存期

与桌面应用程序中的窗体不同,ASP.NET 网页在用户使用窗体时不会启动或运行,并且仅当用户单击"关闭"按钮时才会卸载。这是由于 Web 具有断开连接的特性。浏览器从 Web 服务器请求页面时,浏览器和服务器相连的时间仅够处理请求。Web 服务器将页面呈现到浏览器之后,连接即终止。如果浏览器对同一 Web 服务器发出另一个请求,则即使是对同一个页面发出的,该请求仍会作为新请求来处理。

Web 这种断开连接的特性决定了 ASP.NET 页的运行方式。用户请求 ASP.NET 网页时,将创建该页的新实例。该页执行其处理,将标记呈现到浏览器,然后该页被丢弃。如果用户单击按钮执行回发,将创建该页的新实例;该页执行其处理,然后再次被丢弃。这样,每个回发和往返行程都会导致生成该页的一个新实例。

ASP.NET 页运行时,此页将经历一个生命周期,在生命周期中将执行一系列处理步

骤。这些步骤包括初始化、实例化控件、还原和维护状态、运行事件处理程序代码以及进行呈现。了解页的生命周期非常重要,这样就能在合适的生命周期阶段编写代码,以达到预期效果。此外,如果开发自定义控件,则必须熟悉页生命周期,从而可以正确地初始化控件,使用视图状态数据填充控件属性以及运行所有控件行为逻辑。控件的生命周期基于页的生命周期,但是页引发的控件事件比单独的 ASP.NET 页中可用的事件多。

一般来说,页要经历表 1-1 所示的各个阶段。除了页生命周期阶段以外,还有在请求前后出现的应用程序阶段,但是这些阶段并不特定于页。

表 1-1 页面生存期

阶 段	说 明
页请求	页请求发生在页生命周期开始之前。用户请求页时,ASP.NET 将确定是否需要分析和编译页(从而开始页的生命周期),或者是否可以在不运行页的情况下发送页的缓存版本以进行响应
开始	在开始阶段,将设置页属性,如 Request 和 Response。在此阶段,页还将确定请求是回发请求还是新请求,并设置 IsPostBack 属性。此外,在开始阶段期间,还将设置页的 UICulture 属性
页初始化	页初始化期间,可以使用页中的控件,并将设置每个控件的 UniqueID 属性。此外,任何主题都将应用于页。如果当前请求是回发请求,则回发数据尚未加载,并且控件属性值尚未还原为视图状态中的值
加载	加载期间,如果当前请求是回发请求,则将使用从视图状态和控件状态恢复的信息加载控件属性
验证	在验证期间,将调用所有验证程序控件的 Validate 方法,此方法将设置各个验证程序控件和页的 IsValid 属性
回发事件处理	如果请求是回发请求,则将调用所有事件处理程序
呈现	在呈现期间,视图状态将被保存到页,然后页将调用每个控件,以将其呈现的输出提供给页的 Response 属性的 OutputStream
卸载	完全呈现页、将页发送至客户端并准备丢弃时,将调用卸载。此时,将卸载页属性(如 Response 和 Request)并执行清理

6. 生命周期事件

在页生命周期的每个阶段中,页将引发可运行用户代码进行处理的事件。对于控件事件,通过声明方式使用属性(如 onclick)或使用代码的方式,均可将事件处理程序绑定到事件。

页还支持自动事件连接,即 ASP.NET 将寻找具有特定名称的方法,并在引发特定事件时自动运行这些方法。如果@ Page 指令的 AutoEventWireup 属性设置为 true(或者如果未定义该属性,因为默认情况下为 true),页事件将自动绑定至使用 Page_event 命名约定的方法,如 Page_Load 和 Page_Init。

表 1-2 列出了常用的页生命周期事件。实际的事件比列出的事件要多。但是,它们不用于大多数页处理方案,而是主要由 ASP.NET 网页上的服务器控件使用,以初始化和呈现它们本身。如果要编写自己的 ASP.NET 服务器控件,则需要详细了解这些阶段。

表 1-2 页生命周期事件

事　件	典 型 使 用
Page_PreInit	使用 IsPostBack 属性确定是否是第一次处理该页； 创建或重新创建动态控件； 动态设置主控页； 动态设置 Theme 属性； 读取或设置配置文件属性值
Page_Init	读取或初始化控件属性
Page_Load	读取和更新控件属性
Control events	执行特定于应用程序的处理： 如果页包含验证程序控件，在执行任何处理之前检查页和各个验证控件的 IsValid 属性； 处理特定事件，如 Button 控件的 Click 事件
Page_PreRender	对页的内容进行最后更改
Page_Unload	执行最后的清理工作，可能包括： 关闭打开的文件和数据库连接； 完成日志记录或其他特定于请求的任务

7. 保留页面状态

在标准的 HTTP 协议中，服务器所拥有的与页面有关的所有信息就是用户使用页面上的控件所指定的信息，因为在发送页面时，浏览器仅会将这些信息发送给服务器。其他信息(例如变量值和属性设置)将被放弃。ASP.NET 通过下列方式帮助保留其他页面信息：

(1) ASP.NET 会在往返行程间保存控件设置(属性)，这种行为称为保存控件状态。

(2) ASP.NET 提供状态管理功能，以便可以在往返行程间保存自己的变量和应用程序特定或会话特定的信息。

(3) ASP.NET 可以检测何时第一次请求页面以及何时发送页面，这使用户可以相应地进行编程。例如，可能想要在第一次显示页面(而不是在每次回发)时从数据库读取信息。

8. ASP.NET 网页的编程

可以使用.NET Framework 支持的各种语言(包括 Visual Basic、C# 和 J#)为 ASP.NET 网页创建服务器代码。ASP.NET 网页可以包含在浏览器中运行的客户端脚本。某些 ASP.NET 功能会生成客户端脚本，并将其插入到页面中。在这种情况下，ASP.NET 始终会生成 ECMAScript (JavaScript)，以实现最佳跨浏览器功能。此外，还可以添加客户端脚本以实现自定义功能。如果这样做，便可以使用与目标浏览器兼容的任何客户端脚本语言。

9. 服务器控件

如同其他网页，ASP.NET 网页可以包含静态文本。ASP.NET 网页通常会向页面添加控件，例如文本框、复选框和按钮。用户可以使用这些控件与页面交互，并在发送回页面时将信息发送到服务器。

ASP.NET 提供一个控件集合，称为 Web 服务器控件。ASP.NET 服务器控件可能类

似于对应的 HTML 窗体元素。例如,ASP.NET 的 TextBox 控件类似于 HTML <input type="text"> 标记。但是,ASP.NET 服务器控件拥有比 HTML 元素更为广泛的功能。可以在 ASP.NET 页上使用的服务器控件包括:日历控件、显示列表或表格的数据绑定控件、用于向站点添加安全性的登录控件以及其他更多控件。

10. 页面和服务器控件事件

ASP.NET 网页以及网页上的控件支持事件模型。例如,用户单击 ASP.NET 网页上的 Button 服务器控件时,页面会发回到服务器,然后重新创建该页面,并引发一个 Click 事件。可以将代码添加到响应此 Click 事件的页面。

初始化页面时,页面本身会引发生命周期事件(如 Page_Init 和 Page_Load 事件),这为用户提供了在启动页面时运行代码的机会,每次往返行程都会创建页面并重新初始化页面,每个控件都会引发它自己的事件。按钮控件引发一个 Click 事件,复选框和单选按钮控件引发一个 CheckedChanged 方法,而列表框和下拉列表控件则引发一个 SelectedIndexChanged 事件。某些控件(如 Calendar 控件)引发的事件比简单的 Click 事件更抽象。例如,用户导航至不同的月份时,Calendar 控件引发一个 VisibleMonthChanged 事件。

大多数 ASP.NET 服务器控件仅支持很少的几个可以在服务器代码中处理的事件。要处理某个事件,页面必须执行一次往返行程,以便用户的选择可以发送到页面以待处理。服务器控件不公开诸如 onmouseover 之类的高频率事件,因为每次引发此类事件时,都会发生到服务器的另一个往返行程,这将显著影响页面中的响应时间。但是,可以将 ASP.NET 服务器控件配置为引发客户端事件,例如 onmouseover。在这种情况下,控件不发送回服务器,因而由用户创建客户端脚本来响应事件。

1.4 项 目 实 施

下面将讨论简单网上书店 Web 网站创建过程,项目实施过程分解为如下 3 个任务:配置 Web 服务器 IIS;使用 Visual Studio 2005 创建简单网站页面;使用 IIS 发布 Web 应用程序。每个任务由若干步完成。

1.4.1 任务 1-1 配置 Web 服务器 IIS

1. 任务介绍

了解了整个 .NET 架构后,将使用 ASP.NET 的开发平台开发 Web 应用程序网站。ASP.NET 应用程序网站通过 IIS 发布,本任务将介绍 Web 服务器 IIS 的配置。

2. 任务分析

要建立 ASP.NET 平台需要的软件如下:

- Windows 操作系统
- IIS 6.0

- Net Framework SDK
- IE 浏览器

一般计算机上都已经安装了 Windows 操作系统、IE 浏览器，Net Framework SDK 在安装 Visual Studio 2005 集成开发环境时会自动安装，因此这里只介绍一下 IIS 的安装步骤。

(1) 配置 Web 服务器 IIS

要成为网站服务器，只要有 IIS(Internet Information Services)的服务程序即可。IIS 最主要的功能有以下几个：

① 响应使用者的要求，将所要浏览的网页内容传输给他们。

② 管理及维护 Web 站点。

③ 管理及维护 FTP 站点。

④ SMTP(Simple Mail Transfer Protocol)虚拟服务器。

目前，用得比较多的版本是 IIS 6.0，它是 Windows 2003 的内建组件，除了 Professional 需使用控制台的“新增/移除程序”另外安装到系统中以外，Server 等其他版本在安装 Windows 2003 后就已经在系统内提供服务了，所以要建立 ASP.NET 开发平台，使用 Windows Server 2003 比较方便。若 Windows Server 2003 中没有 IIS 6.0，请按下列步骤安装：

① 选择“开始”→“设置”→“控制面板”选项。

② 单击“添加或移除程序”按钮，并选择“新增/移除 Windows 组件”选项。

③ 出现图 1-1 所示窗口后，选择“应用程序服务器”选项。

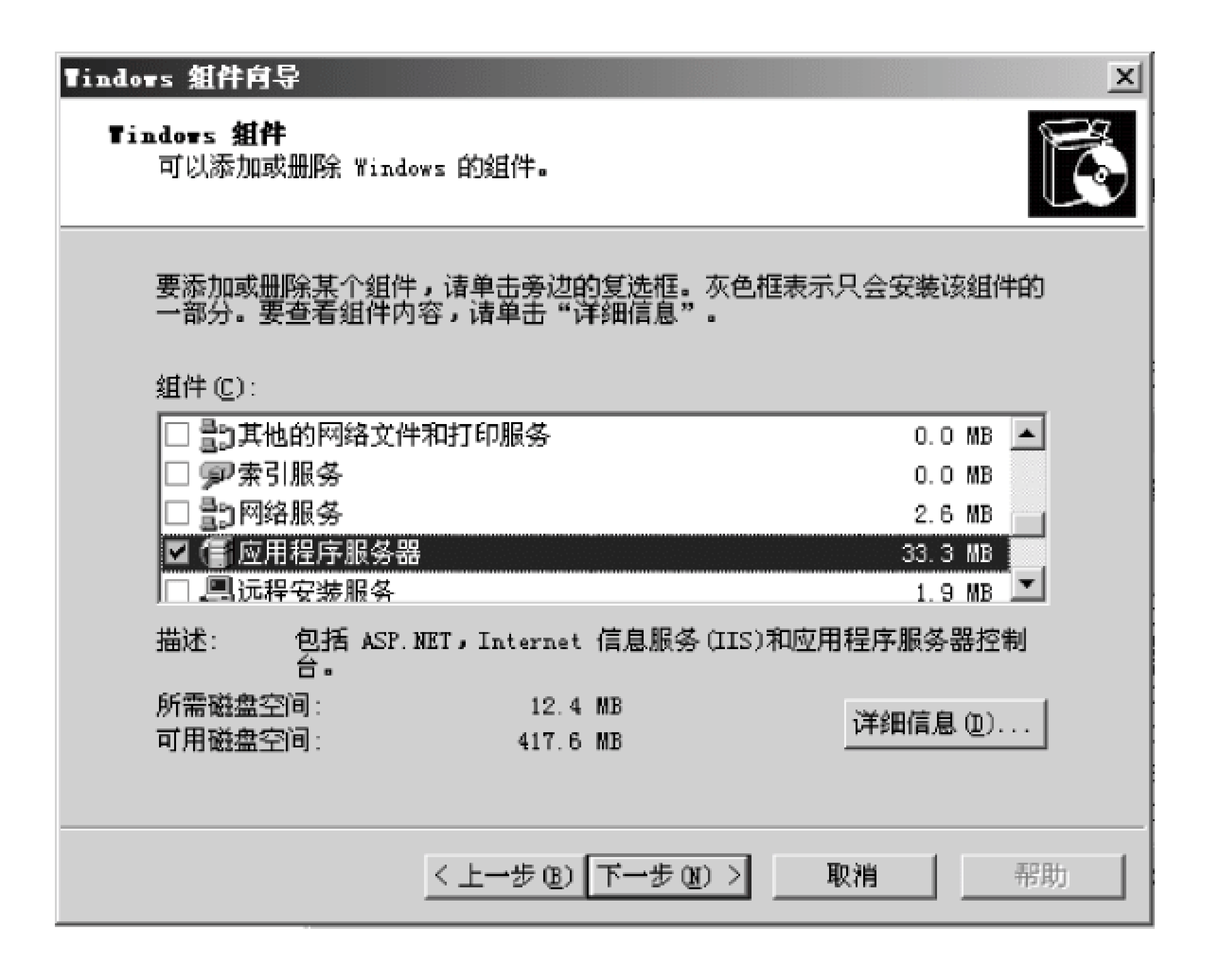


图 1-1　新建母版页窗口

④ 单击“下一步”按钮出现图 1-2 所示窗口后，选择“Internet 信息服务(IIS)”选项。

⑤ 单击“确定”按钮即可完成 IIS 6.0 的安装。

(2) 常见问题解决方案

虽然采用 IIS 6.0 配置 Web 服务比较简单，可有时还是会或多或少地出现问题。以下是对两个常见问题的总结。

① HTTP 错误 404-文件或目录未找到。

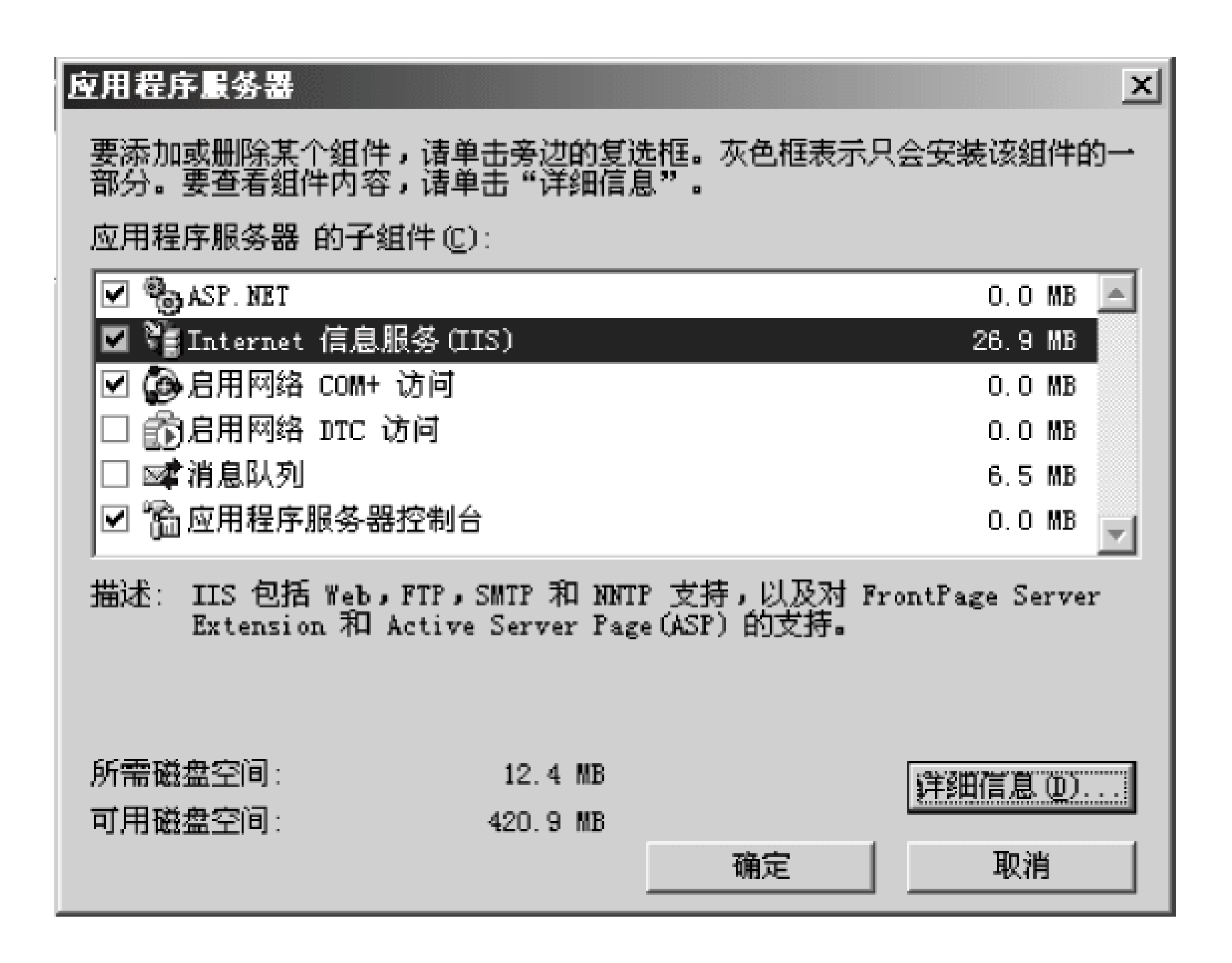


图 1-2 IIS 组件选择界面

分析解决：此类问题十分常见。原因是在 IIS 6.0 中新增了"Web 程序扩展"这一项，而里面的很多服务在默认情况下都是禁止的。直接在"Web 程序扩展"中启用"Active Server Pages"即可。

② HTTP 错误 401.2-未经授权访问由于服务器配置被拒绝。

分析解决：造成此类问题的原因应该是身份验证设置的问题，一般将其设置为匿名身份认证即可，这是大多数站点使用的认证方法。

3. 任务完成总结

通过这个任务，我们了解了 IIS 的配置过程，并讨论了常见问题的解决方法。

4. 课堂训练与拓展

依据本任务中介绍的步骤，配置 IIS 服务器。

1.4.2 任务 1-2 使用 Visual Studio 2005 创建简单 Web 网站页面

1. 任务介绍

设计一个简单的网站，要求在登录页面的文本框里输入姓名，系统能返回一个问候信息。

2. 任务分析

在这个任务中将介绍 Visual Studio 2005 集成开发环境，并且按照一般网站开发的次序，介绍 Visual Studio 2005 中是如何实现简单页面设计的。通过示例，可以发现 Visual Studio 2005 控件丰富，功能强大，可以帮助用户快速开发网站。

(1) 创建网站

Visual Studio 2005 集成开发环境作为实现 ASP.NET 技术的 IDE 集成开发环境，在各

个方面给予 ASP.NET 技术支持，提供了许多功能来协助创建网站。如果能充分利用这些功能，必将能大幅度提高网页的开发效率。

首先，需要安装 Visual Studio 2005，安装方式非常简单，按照向导提示安装即可，安装时选择全部的功能。如果没有光盘，也可以通过下载安装 Visual Web Develop 2005 Express。

下面介绍创建网站的方法。

① 选择"文件"→"新建"→"网站"命令，或者单击"起始页"→ "网站(W)…"按钮，如图 1-3 所示。



图 1-3 打开新建网站窗口

② 系统将出现"新建网站"对话框，如图 1-4 所示。首先在这个窗口中选择"ASP.NET 网站"，然后在"位置"输入框中输入新建网站的路径，在"语言"选择栏里选择使用的语言 Visual C＃，最后单击"确定"按钮创建网站。

③ 创建网站以后，系统会创建一个 WebSite1 文件夹，默认情况下，该文件夹包含一个 App_Data 子文件夹和一个默认网页 Default.aspx，如图 1-5 所示。

(2) Visual Studio 2005 开发环境

在新建网站后，可以看到 Visual Studio 2005 开发环境是由很多窗口构成的，下面介绍常用窗口的使用方法。

① 工具箱窗口。选择"视图"→"工具箱"命令，或单击"工具箱"快捷按钮。由于 ASP.NET 控件比较多，所以按组进行分类，单击"＋"按钮可以展开。如图 1-6 所示展开的是验证组里的验证控件。

工具箱控件功能很多，开发网站时尽量使用这些控件来完成，以保证网站的安全性、可

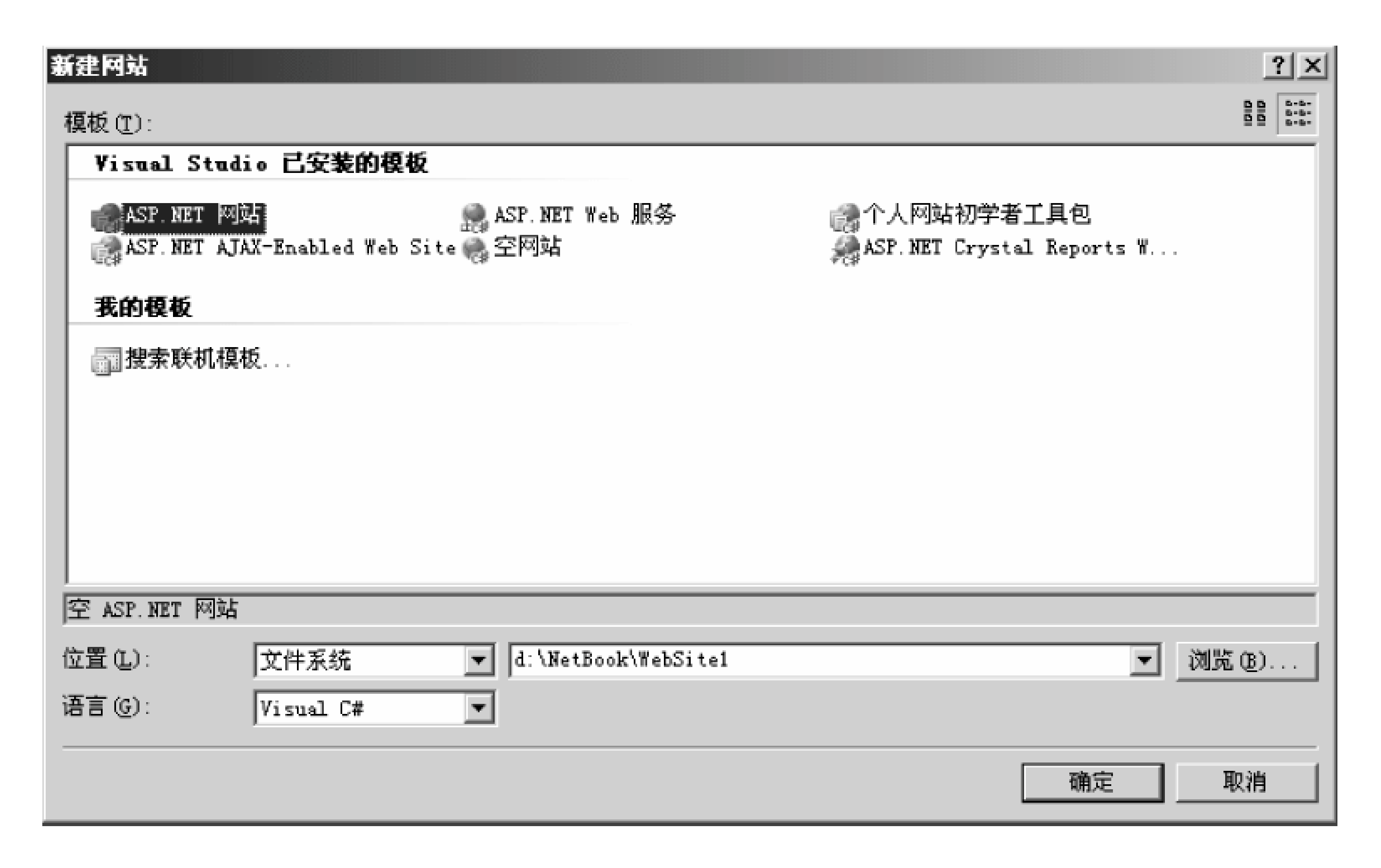


图 1-4 “新建网站”对话框

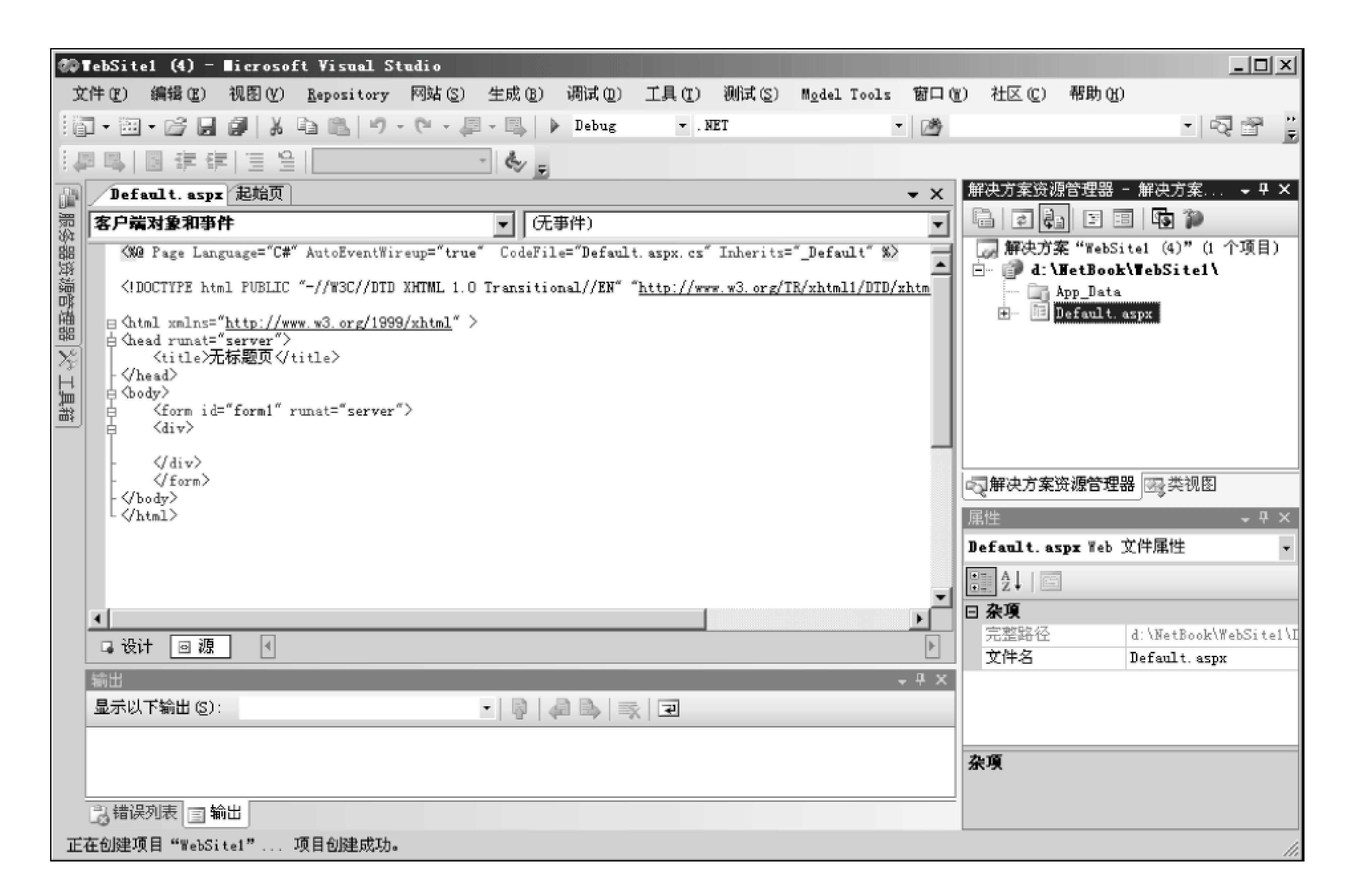


图 1-5 成功新建网站

靠性和高效性。

除了常用的工具箱以外，ASP.NET 还有很多没有列出来的控件，甚至允许用户导入第三方的控件，其步骤如下：

选择“工具”→“选择工具箱项”菜单项或者在工具箱上右击，在弹出的菜单中单击“选择项”菜单项，如图 1-7 所示。

在打开的“选择工具箱项”窗口中，在需要添加的控件前打勾，就会在当前的工具箱列表中添加相应的控件。如果选择“浏览”按钮，还可以导入任何以 .dll、.ocx、.exe 为扩展名的

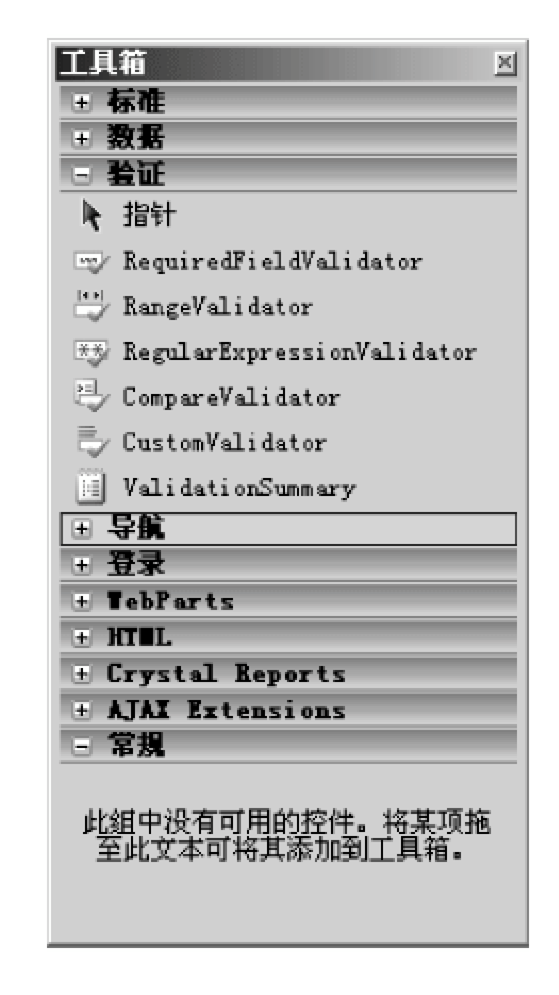


图 1-6　展开工具箱



图 1-7　“选择项”命令

第三方控件，如图 1-8 所示。

② 解决方案资源管理器窗口。选择“视图”→“解决方案资源管理器”命令，或单击“解决方案资源管理器”快捷按钮，打开“解决方案资源管理器”窗口，如图 1-9 所示。



图 1-8　“选择工具箱项”窗口



图 1-9　“解决方案资源管理器”窗口

在“解决方案资源管理器”窗口中选中创建的网站，右击，弹出快捷菜单，如图 1-10 所示。该菜单主要是对网站项目进行管理，例如文件的管理、引用的管理、启动项的管理等，相关菜单用途会在以后的项目中陆续介绍。

③ 属性窗口。选择“视图”→“属性窗口”命令，或单击“属性窗口”快捷按钮，打开“属性”窗口，如图 1-11 所示。该窗口的下拉菜单在页面无法选择相应控件时可以起到很好的作用，许多人习惯用鼠标在页面上选中控件，然后设置属性，当有的控件无法用鼠标在页面上选择时，可以采用该方式选择。

④ 页面编辑窗口。页面编辑窗口是开发人员使用频繁的窗口之一，ASP.NET 2.0 提供了两种编辑网页的模式，即设计模式和源模式。在设计模式下，可以将工具箱内的控件通




图 1-10　打开快捷菜单

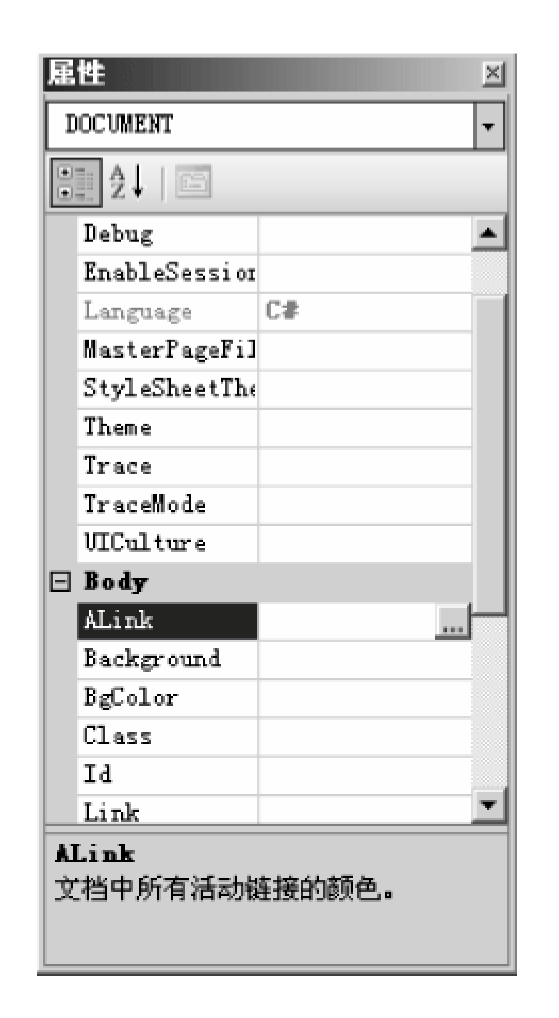


图 1-11　“属性”窗口

过拖曳的方式直接放到页面上去，然后通过单击控件设置其属性。在源模式下，可以直接通过写 HTML 代码的方式完成网页编辑工作，这是一种纯代码的方式。一般在开发过程中使用设计模式，除非需要手动添加代码时才在源模式下操作。这两种编辑模式本质上是一样的，但是如果在源模式下编写的 HTML 语句有错误时，将无法切换到设计模式，如图 1-12 所示。

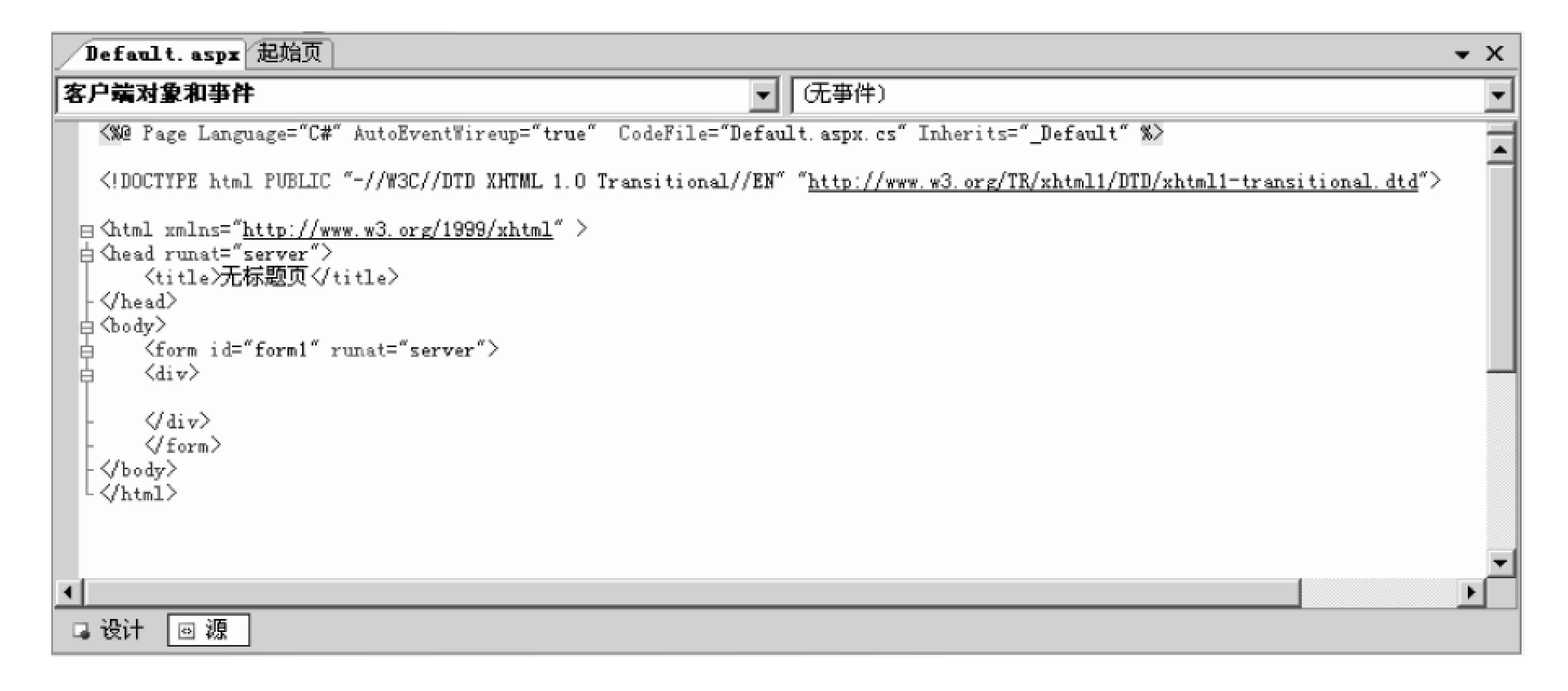


图 1-12　源模式编辑窗口

⑤ 服务器资源管理器窗口。选择“视图”→“服务器资源管理器”，或单击“服务器资源管理器”快捷按钮，打开服务器资源管理器，如图 1-13 所示。该窗口主要实现数据连接相关的功能。

(3) 添加新网页

前面介绍了如何新建并打开一个网站，一个网站的页面往往不止一页，下面将介绍如何向已存在的网站中添加新的页面。

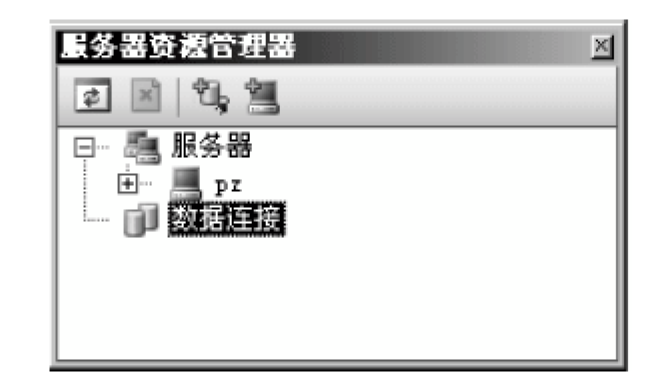


图 1-13　服务器资源管理器

① 添加新的网页。在“解决方案资源管理器”窗口中，右击网站文件夹，从弹出的快捷菜单中选择“添加新项”命令，或选择“文件”→“新建”→“文件”命令添加新的网页。

② 设置网页。在打开的如图 1-14 所示窗口中，可以发现有很多文件类型，以后将陆续介绍这些文件类型的用途和使用方法，在这里选择“Web 窗体”，并将新建的网页起名为“login.aspx”。



图 1-14　添加新的 Web 窗体

③ 打开新创建的网页。创建新的网页后，可以在解决方案资源管理器中发现该网页，这时网页编辑窗口已经把当前页面显示在窗口中，并默认为源模式。单击“+”，可以看到该页面分为两个部分，一部分是以.aspx 为扩展名的页面设计部分；另一部分是以.aspx.cs 为扩展名的页面代码部分。这是因为在前面创建网页的时候选择了“将代码放在单独的文件中”这个选项。ASP.NET 使用页面设计与代码相分离的模式，极大地方便了网页的制作。一般来讲，网页制作包括界面设计和编写代码实现业务流程两个部分。在 ASP.NET 中，可以把精力集中在编写代码实现业务流程上，而界面设计和代码美化可以交给美工处理。使用这种分离模式，可以使得美工和程序员很好地合作完成网页制作，而不必像以前那样必须等界面设计好才可以编写代码。

(4) 编辑、运行页面

将页面编辑窗口切换到设计模式，可以将工具箱上的控件拖曳到设计窗口，快速创建控件，步骤如下。

① 设计一个登录界面，如图 1-15 所示。这个界面可以通过“工具箱”中的 HTML 元素在设计模式下直接设计，也可以先利用其他网页设计工具设计好网页后，将网页源代码复制到这个页面中。

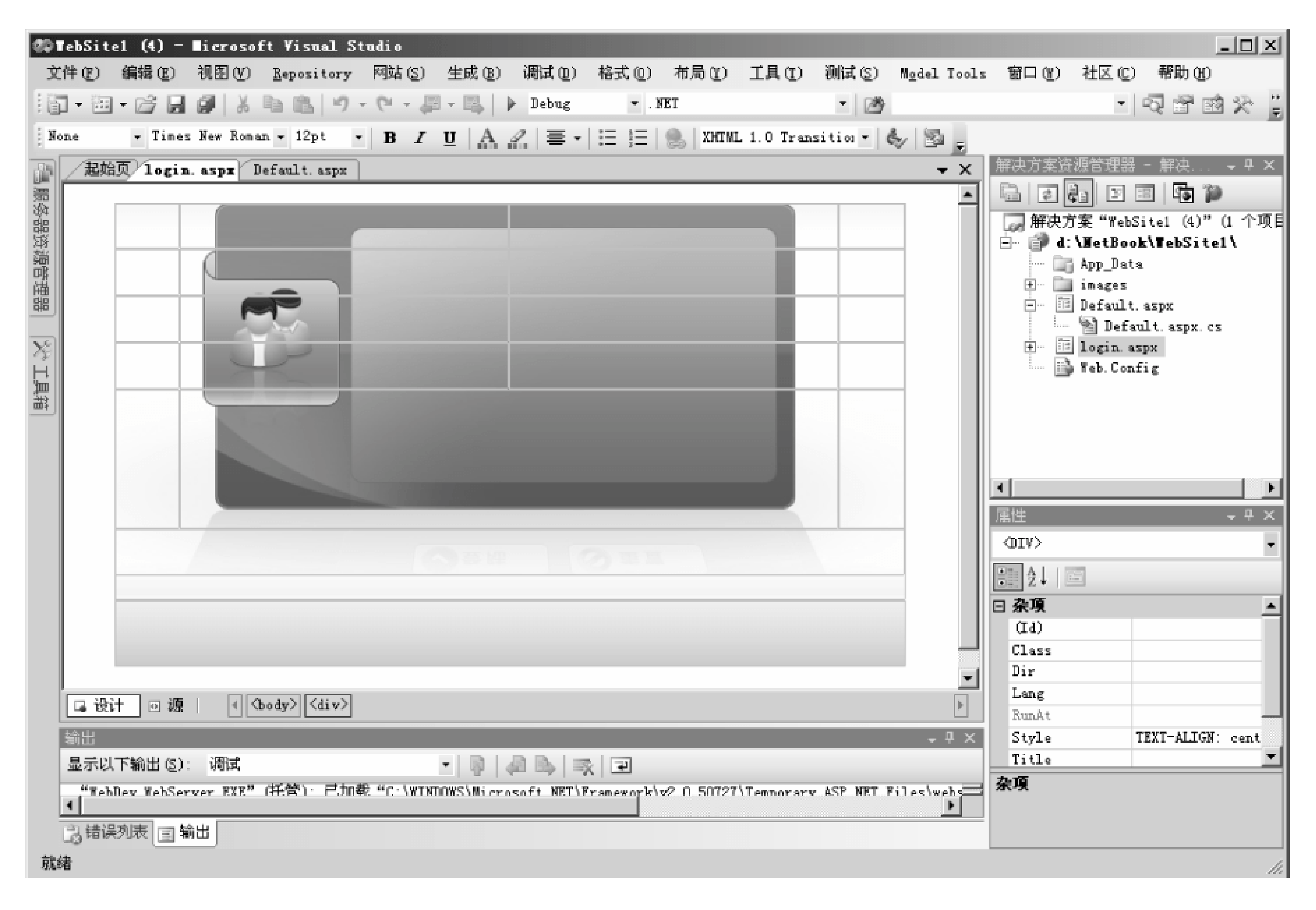


图 1-15　添加新的 Web 窗体

② 将工具箱上的控件拖曳到设计窗口，快速创建控件。例如，拖曳 Label、TextBox、ImageButton 控件到 login.aspx 设计窗口，如图 1-16 所示。

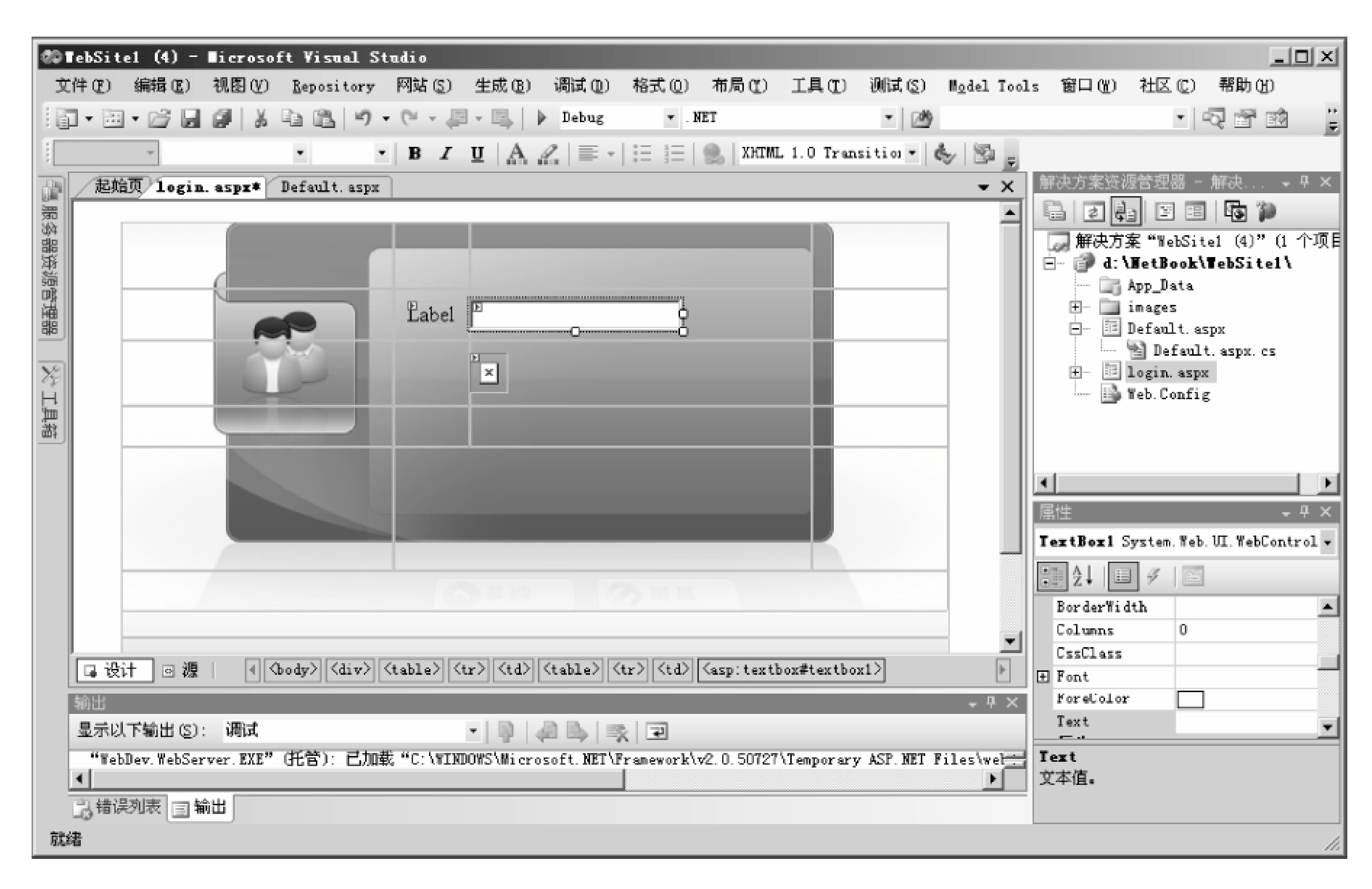


图 1-16　拖曳控件至设计窗口

③ 在“属性”窗口可设置控件属性。修改 Label1 的 Text 属性为“姓名：”；修改 ImageButton1 控件的 ImageUrl 属性为“～/images/btn_login. gif”，如图 1-17 所示。

除了在“属性”窗口中修改值外，也可以在源模式下编辑网页，如图 1-18 所示。

④ 选择“调试”→“启动调试”命令或按 F5 键运行网页。

首先，必须单击要运行的网页，可以直接单击“文件”窗口上方的标签，或在“解决方案资源管理器”窗口中单击要运行的网页。然后，选择“调试”→“启动调试”命令，或按 F5 键，如图 1-19 所示。



图 1-17　ImageButton1 的“属性”窗口

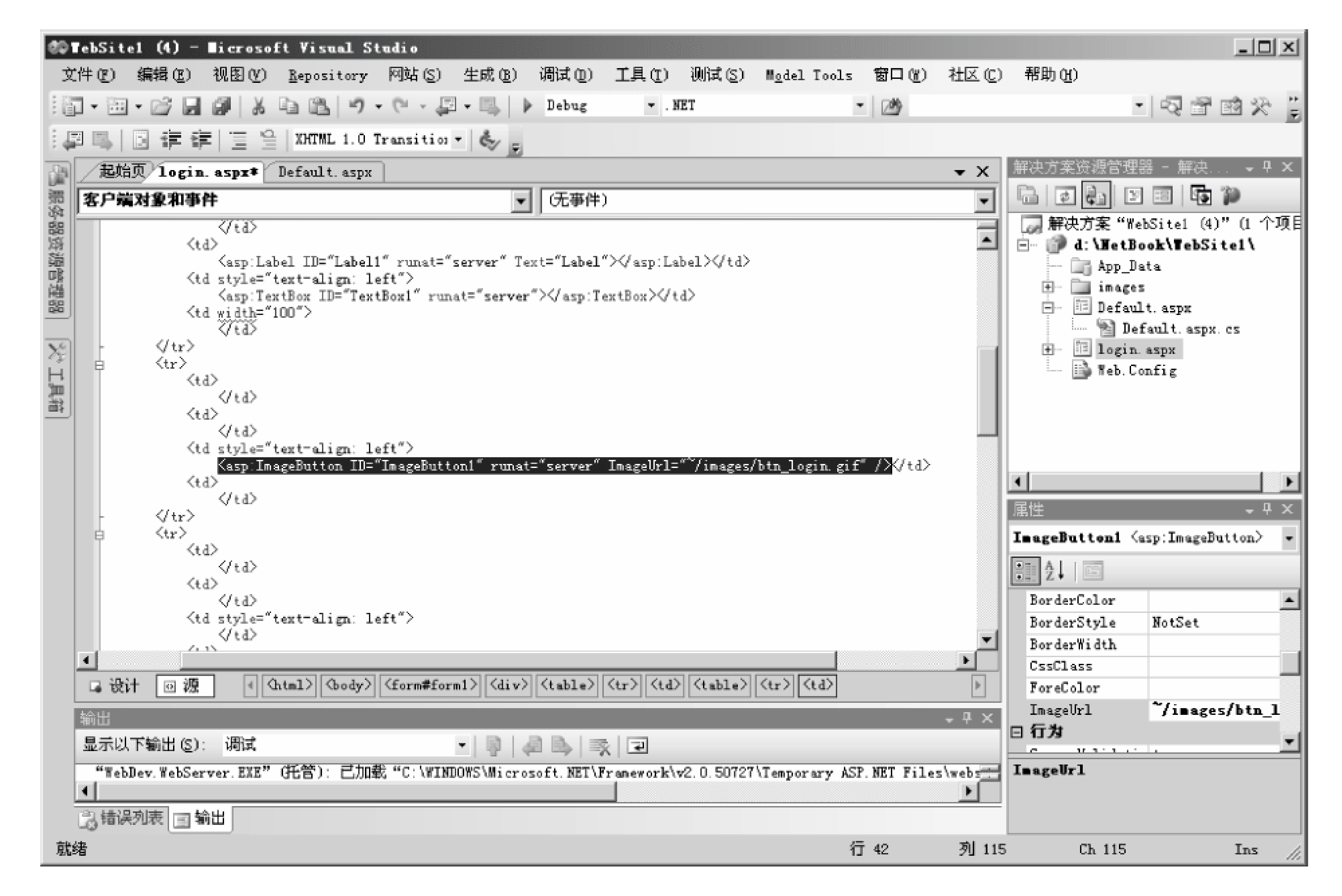


图 1-18　源模式窗口

⑤ 设置调试功能。当第一次运行时，系统会询问是否在 Web. config 中加入调试功能，如图 1-20 所示，单击“确定”按钮即可。

⑥ 运行后的结果如图 1-21 所示。

(5) 添加事件代码

① 添加按钮单击事件有两种创建控件事件的方法，下面分别进行介绍。

方法 1：这是最简单的方法，双击所要创建事件的控件，即可创建 Click 事件。

方法 2：方法 1 通常只能创建 Click 事件，假设要创建其他更多的事件，必须在控件“属性”窗口创建事件。如图 1-22 所示，利用 ImageButton1 的属性窗口创建事件。

如图 1-23 所示为在代码文件中创建的 ImageButton1_Click 事件。

② 输入 ImageButton1_Click 事件代码。此代码的功能是当用户单击按钮时，在用户输入的文字前加上“欢迎”，并显示在消息框上。

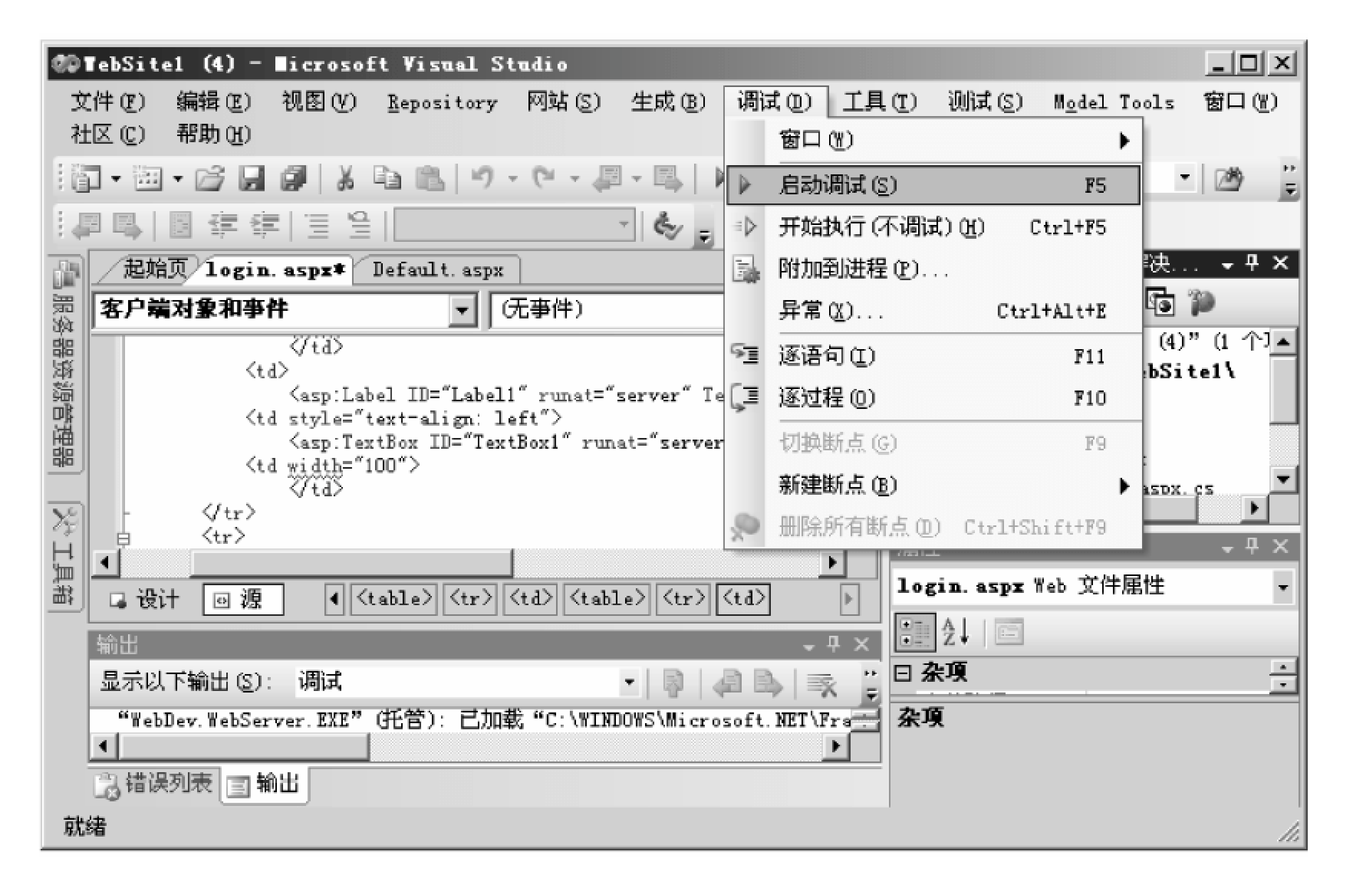


图 1-19 在 IE 浏览器运行网页

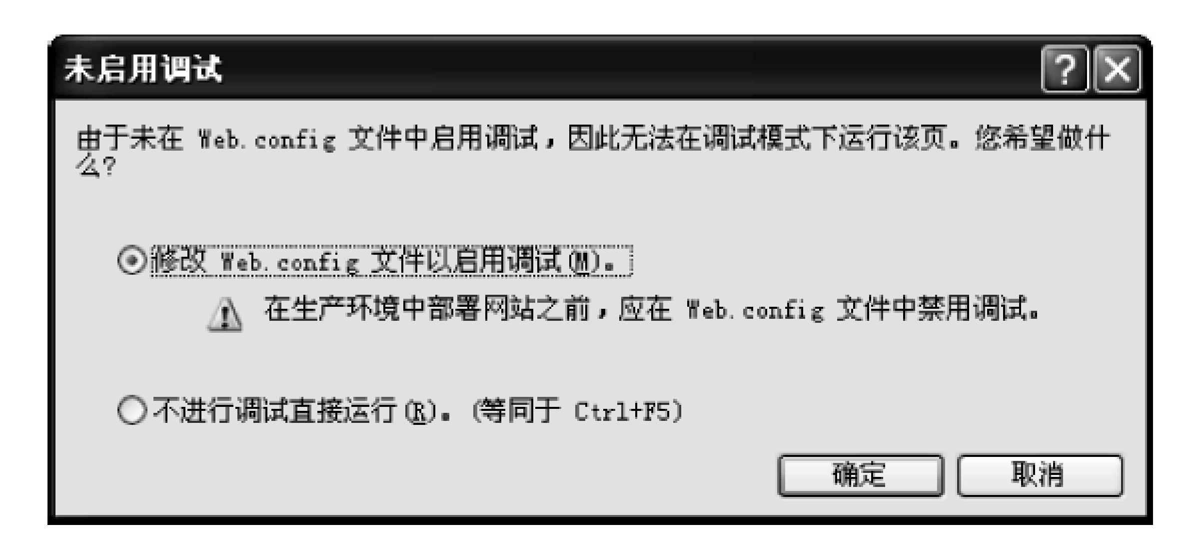


图 1-20 设置调试选项



图 1-21 运行后的结果

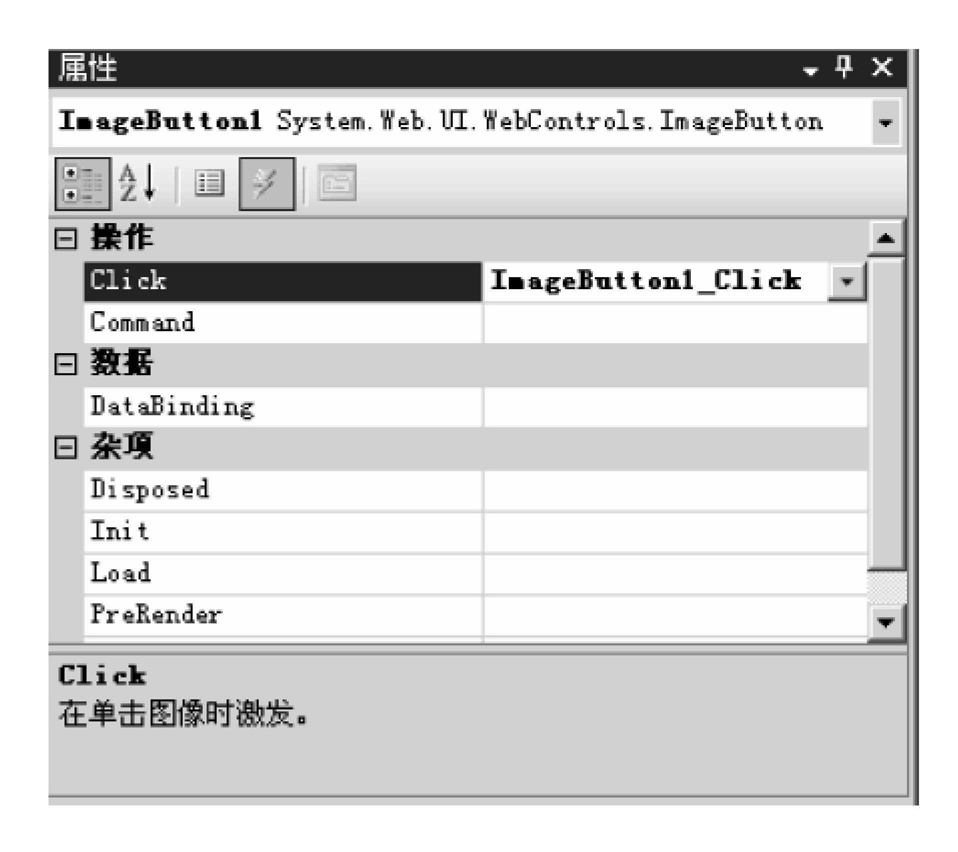


图 1-22　利用“属性”窗口创建事件

```
login.aspx.cs* 起始页 login.aspx*
login    ImageButton1_Click(object sender, ImageClickEver
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {

    }
}
```

图 1-23　ImageButton1_Click 事件

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    string s = "欢迎" + TextBox1.Text;
    Response.Write("<script>alert('" + s + "')</script>");
}
```

(6) 运行结果

按 F5 键运行代码后的窗口如图 1-24 所示。示例中是在 TextBox 中输入“张三”,然后单击“登录”按钮,将显示“欢迎张三”的消息框,如图 1-25 所示。

3. 任务完成总结

通过这个任务,主要学习了 Visual Studio 2005 集成开发环境,并学习了使用集成开发环境设计一个简单网页的过程。熟悉整个开发环境对于后期的程序开发起着至关重要的作用。



图 1-24 运行窗口



图 1-25 运行结果

4. 课题训练与拓展

依据本任务中的开发模式，设计一个输入年份，能计算出其是否为闰年的网页。

1.4.3 任务 1-3 使用 IIS 发布 Web 应用程序

1. 任务介绍

ASP.NET Web 应用程序需要使用 IIS 发布，用户才能通过网络访问，本任务将介绍使用 IIS 发布任务 1-2 中设计的简单网上书店 Web 应用程序，建立简单的网上书店网站。

2. 任务分析

ASP.NET Web 应用程序可以在 Visual Studio 2005 中直接运行测试，但是当最终发布运行时，需要在脱离 Visual Studio 2005 的环境下使用，因此必须在 IIS 中发布 ASP.NET Web 应用程序。

(1) 在 IIS 中创建 Web 应用程序

① 选择“开始”→“设置”→“控制面板”→“管理工具”→“Internet 信息服务(IIS)管理器”命令，打开“Internet 信息服务(IIS)管理器”窗口。

② 右击“默认网站”选项，在弹出的快捷菜单中选择“新建”→“虚拟目录”命令，如图 1-26 所示。

③ 在弹出的“虚拟目录创建向导”对话框中，输入虚拟目录的名称为“NetBook”，单击“下一步”按钮，如图 1-27 所示。

④ 单击“浏览”按钮，选择在 Visual Studio 2005 中创建的网站项目目录，单击“下一步”按钮，如图 1-28 所示。

⑤“允许下列权限”中有 5 个选项，默认为“读取”选项，再选择“运行脚本(如 ASP)”选

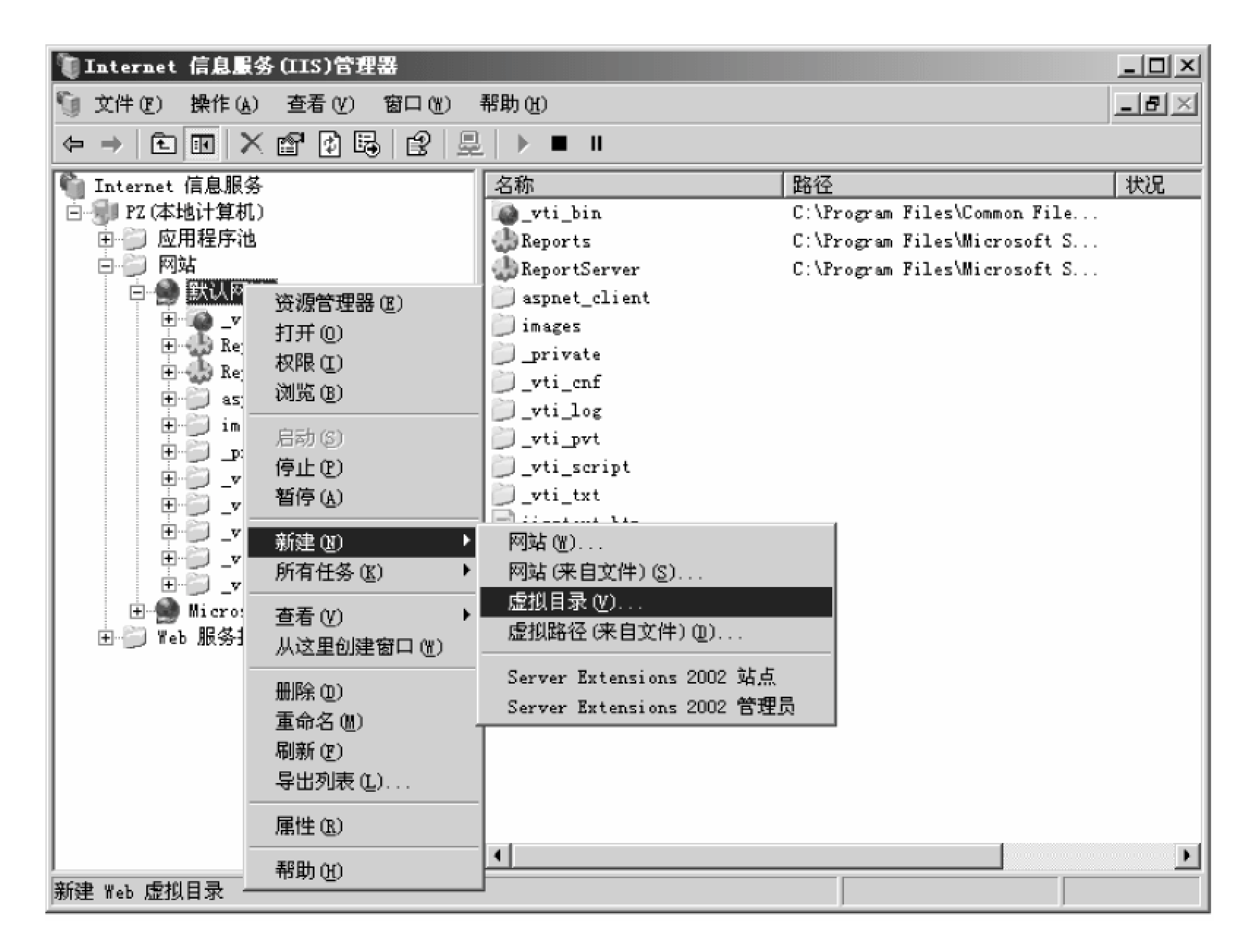


图 1-26　“Internet 信息服务(IIS)管理器”窗口

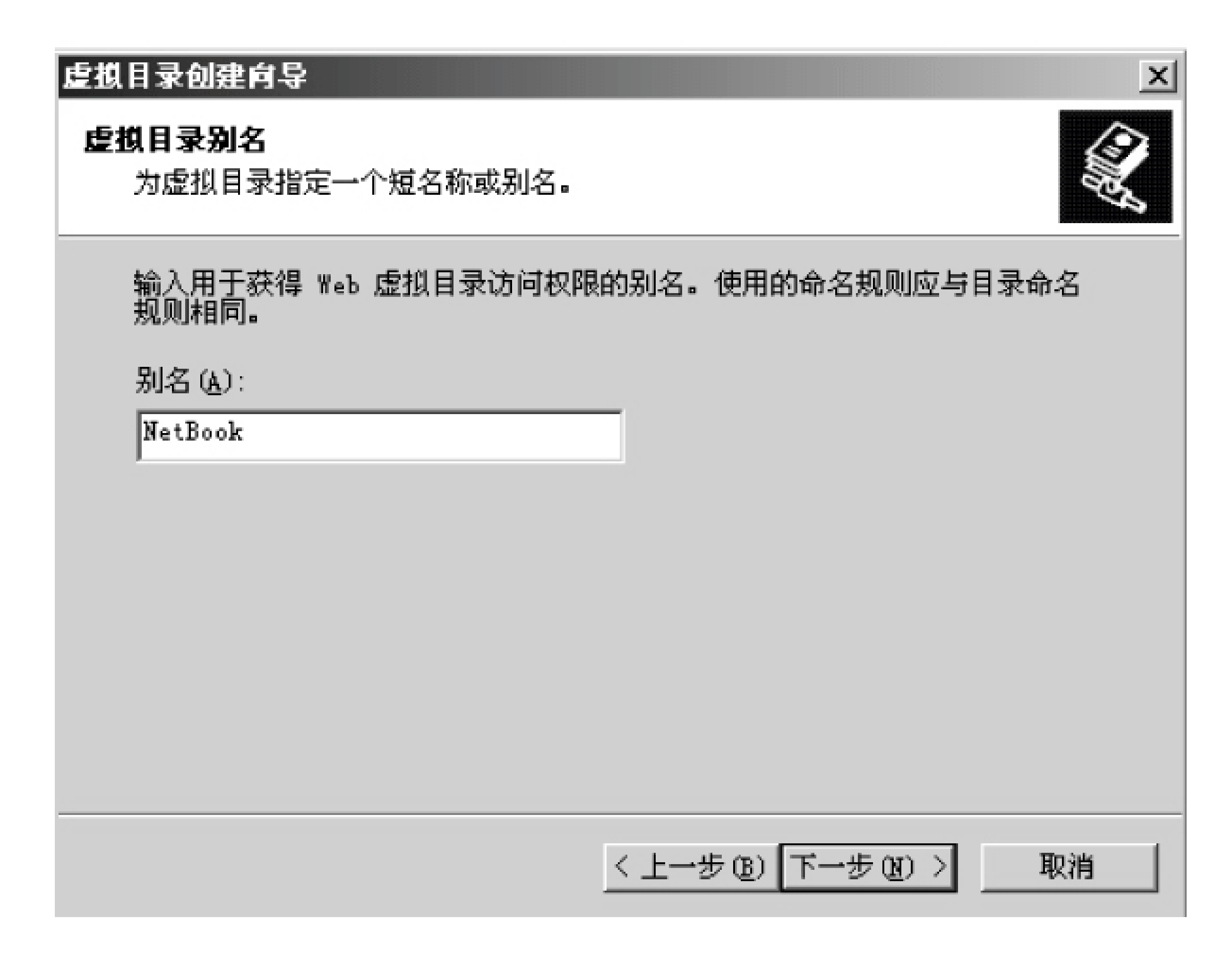


图 1-27　“虚拟目录创建向导”对话框

项，单击“下一步”按钮，如图 1-29 所示。完成后在 IIS 中配置了一个 NetBook 的 Web 应用程序，如图 1-30 所示。

⑥ 选择“NetBook”Web 应用程序，右击，在弹出的菜单中选择“属性”选项。在“属性”对话框中，选择 ASP.NET 选项卡，将“ASP.NET 版本”设置为 2.0，如图 1-31 所示。

(2) 运行发布在 IIS 中的 Web 应用程序

在 IE 浏览器的地址栏里输入“http://localhost/NetBook/login.aspx”，运行创建的 Web 应用程序。

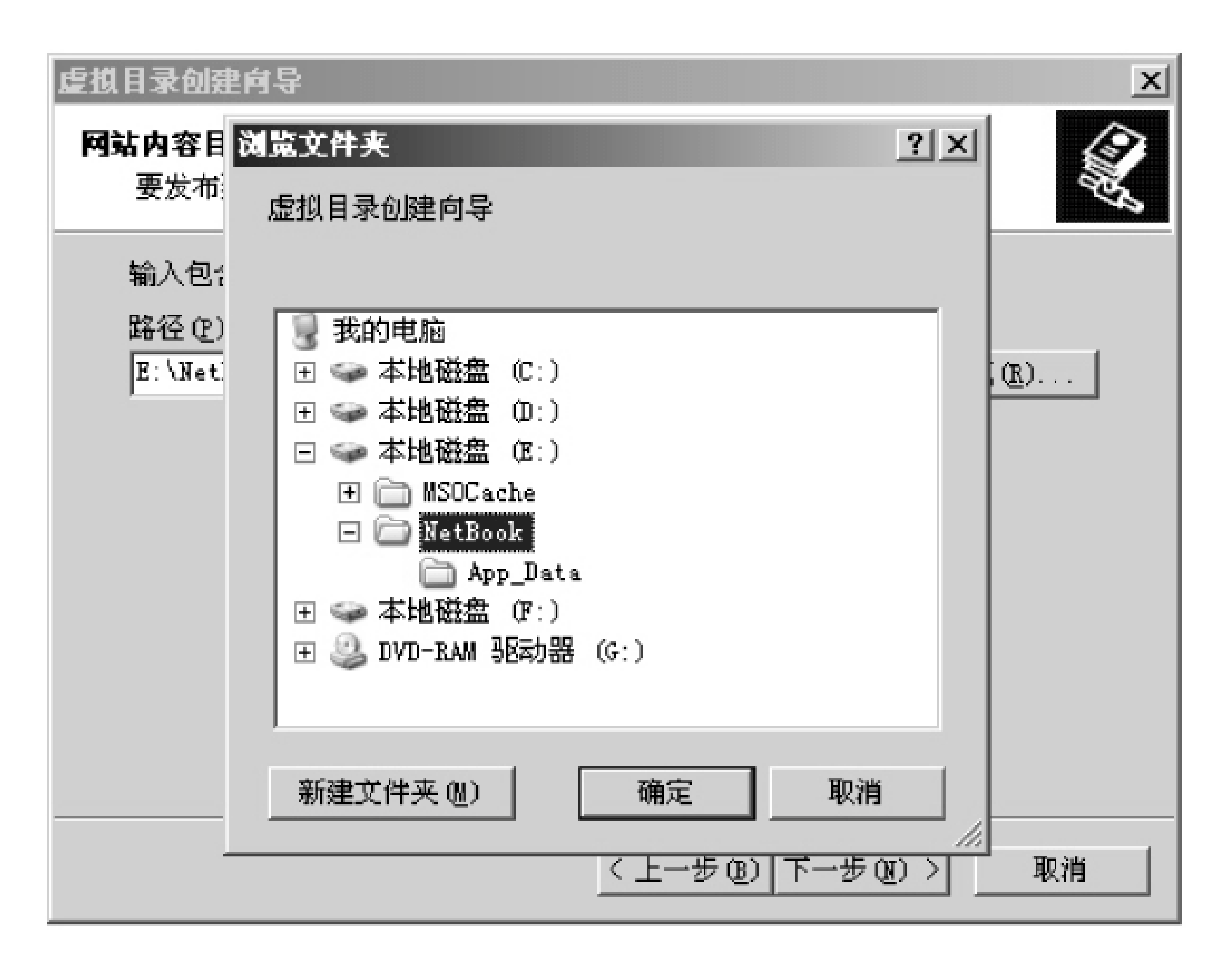


图 1-28 “浏览文件夹”对话框



图 1-29 选择权限

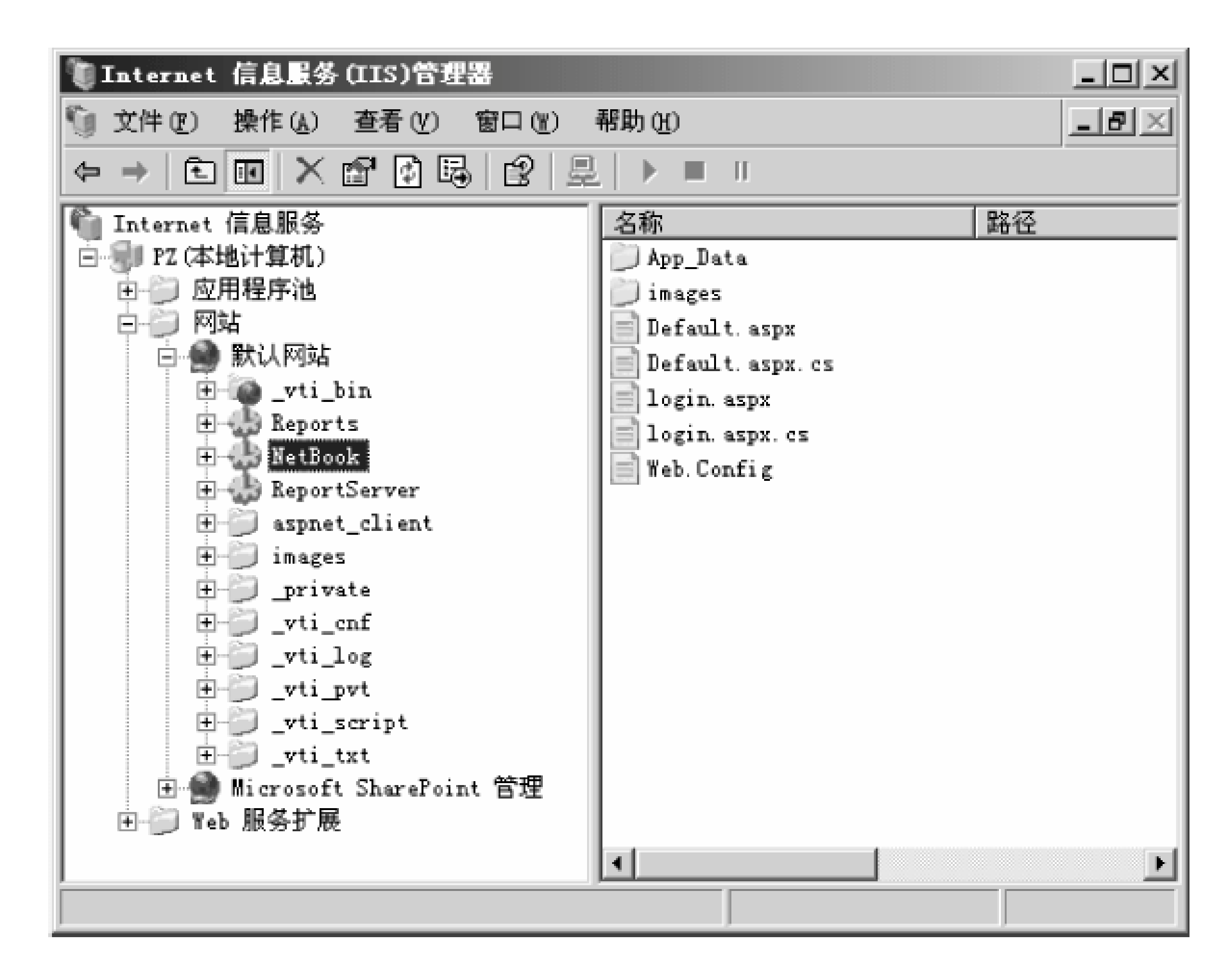


图 1-30 创建好的“NetBook”Web 应用程序



图 1-31　Web 应用程序"属性"对话框

3. 任务完成总结

利用 IIS 发布 ASP.NET 的 Web 应用程序，在操作上和以前发布 ASP 的 Web 应用程序基本没有差异，但是要注意在发布好以后，ASP.NET 的网站有 1.0 版本和 2.0 版本的区别。因此当运行网站时，如果在浏览器中出现"应用程序中的服务器错误"的错误提示，要查看一下是否版本没有设置正确。Visual Studio 2005 中开发的 Web 应用程序使用的是 2.0 版本。

4. 课题训练与拓展

编写一个简单的 Web 应用程序，使用 IIS 发布运行。

1.5　项 目 总 结

在这个项目中介绍了 Visual Studio 2005 集成开发环境，并且按照一般网站开发的次序，介绍了在 Visual Studio 2005 中是如何实现简单页面设计的。Visual Studio 2005 控件丰富，功能强大，可以帮助用户快速开发网站。

1.6　项 目 实 训

1. 任务描述

使用本项目中介绍的方法，创建一个简单的网上书店 Web 网站。

2. 任务要求

① 使用 Visual Studio 2005 集成开发环境创建网上书店页面。
② 使用 IIS 发布网上书店。

项目2

创建风格一致的网上书店Web网站

2.1 项目介绍

项目1介绍了简单网站页面的设计。由于网页一般是由多名开发人员设计的，如果页面设计不统一，网页的界面及风格将各不相同，往往给人的感觉不像是一个整体网站。本项目将介绍设计风格一致、界面美观的网页，并且通过使用母版页、主题、皮肤和样式表等技术维护修改网站整体风格。

2.2 项目分析

统一网站的风格在网站设计中占有重要地位，而网站结构是网站风格统一的重要手段，包括网站布局、文字排版、装饰性元素出现的位置、导航的统一、图片的位置等。使用浏览一些著名的电子商务网站，会发现这些网站在布局和色彩搭配等方面具有统一的风格。这种形式是目前网站普遍采用的设计方法，一方面减少设计、开发的工作量，同时更有利于以后的网站维护与更新。

设计风格一致的网站有多种方式处理，比如在网页设计中用得最多的是级联样式表，通过设立样式表，可以统一地控制HTML中各标记的显示属性。使用级联样式表可以更加有效地控制网页外观，精确指定网页元素位置，以及创建特殊效果。

除了级联样式表以外，本项目中还将重点介绍如何使用ASP.NET中特有的母版页、站点地图、皮肤来设计统一的页面风格。

在实施这个项目前，先学习相关的一些基本知识。

2.3 相关知识

1. 母版

(1) 母版页

使用ASP.NET母版页可以为应用程序中的页面创建风格一致的布局。单个母版页可以为应用程序中的所有页面或一组页面定义所需的外观和标准行为。母版页实际由两部分组成，即母版页本身和内容页。内容页可以是一个或多个。当用户请求内容页时，这些内容

页与母版页合并，将母版页的布局与内容页的内容组合在一起输出。

母版页为具有扩展名.master 的 ASP.NET 文件（如 MySite.master），它是具有可以包括静态文本、HTML 元素和服务器控件的预定义布局文件。母版页由特殊的@Master 指令识别，该指令替换了用于普通.aspx 页的@Page 指令。指令格式如下：

```
<%@ Master Language = "C#" %>
```

@Master 指令可以包含的指令与@Control 指令可以包含的指令大多数是相同的。例如，下面的母版页指令包括一个代码隐藏文件的名称，并将一个类名称分配给母版页。

```
<%@ Master Language = "C#" CodeFile = "MasterPage.master.cs" Inherits = "MasterPage" %>
```

除@ Master 指令外，母版页还包含页的所有顶级 HTML 元素，如 html、head 和 form。例如，在母版页上可以将一个 HTML 表用于布局，将一个 img 元素用于公司徽标，将静态文本用于版权声明并使用服务器控件创建站点的标准导航。可以在母版页中使用任何 HTML 元素和 ASP.NET 元素。

（2）可替换内容占位符

除会在所有页上显示的静态文本和控件外，母版页还包括一个或多个 ContentPlaceHolder 控件。这些占位符控件定义可替换内容出现的区域。接着在内容页中定义可替换内容。定义 ContentPlaceHolder 控件后，母版页代码结构如下所示。

```
<%@ Master Language = "C#" %>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML
    1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml" >
<head runat = "server" >
    <title>Master page title</title>
</head>
<body>
    <form id = "form1" runat = "server">
        <table>
            <tr>
                <td><asp:contentplaceholder id = "Main" runat = "server" /></td>
                <td><asp:contentplaceholder id = "Footer" runat = "server" /></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

（3）内容页

通过创建各个内容页来定义母版页的占位符控件的内容，这些内容页为绑定到特定母版页的 ASP.NET 页（.aspx 文件以及可选的代码隐藏文件）。通过包含指向要使用的母版页的 MasterPageFile 属性，在内容页的@Page 指令中建立绑定。例如，一个内容页可能包含下面的@Page 指令，该指令将该内容页绑定到 Master1.master 页。

```
<%@ Page Language = "C#" MasterPageFile = "Master1.master"  Title = "Content Page" %>
```

在内容页中,通过添加 Content 控件并将这些控件映射到母版页上的 ContentPlaceHolder 控件来创建内容。例如,母版页可能包含名为 Main 和 Footer 的内容占位符。在内容页中,可以创建两个 Content 控件,一个映射到 ContentPlaceHolder 控件 Main,而另一个映射到 ContentPlaceHolder 控件 Footer,如图 2-1 所示。

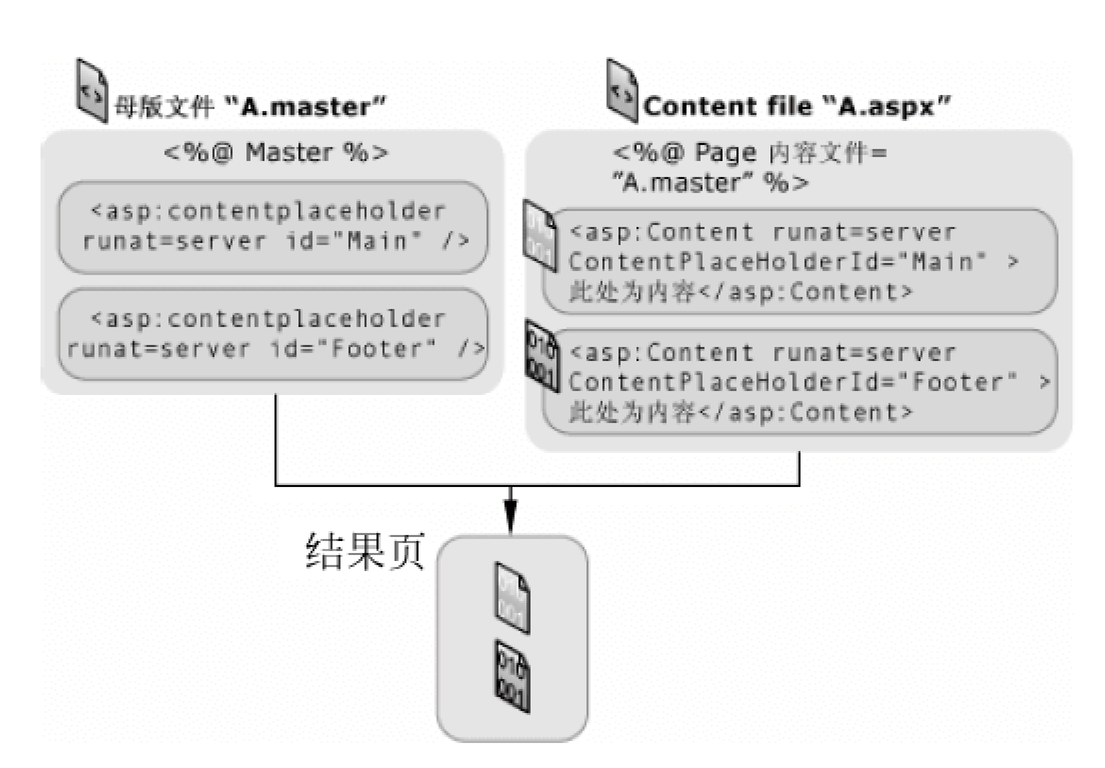


图 2-1 内容页运行机制

创建 Content 控件后,向这些控件添加文本和控件。在内容页中,Content 控件外的任何内容(除服务器代码的脚本块外)都将导致错误。在 ASP.NET 页中所执行的所有任务都可以在内容页中执行。例如,可以使用服务器控件和数据库查询或其他动态机制来生成 Content 控件的内容。

内容页代码结构如下所示。

```
<% @ Page Language = "C#" MasterPageFile = "~/Master.master" Title = "Content Page 1" %>
<asp:Content ID = "Content1" ContentPlaceHolderID = "Main" Runat = "Server">
    Main content.
</asp:Content>
<asp:Content ID = "Content2" ContentPlaceHolderID = "Footer" Runat = "Server" >
    Footer content.
</asp:content>
```

(4) 母版页的优点

母版页提供了开发人员已通过传统方式创建的功能,这些传统方式包括重复复制现有代码、文本和控件元素;使用框架集;对通用元素使用包含文件;使用 ASP.NET 用户控件等。母版页具有下面的优点:

① 使用母版页可以集中处理页的通用功能,以便可以只在一个位置上进行更新。

② 使用母版页可以方便地创建一组控件和代码,并将结果应用于一组页。例如,可以在母版页上使用控件来创建一个应用于所有页的菜单。

③ 通过允许控制占位符控件的呈现方式,母版页使用户可以在细节上控制最终页的布局。

④ 母版页提供一个对象模型,使用该对象模型可以从各个内容页自定义母版页。

(5) 母版页的运行时行为

在运行时,母版页是按照下面的步骤处理的:

① 用户通过输入内容页的 URL 来请求某页。

② 获取该页后，读取@ Page 指令。如果该指令引用一个母版页，则也读取该母版页。如果这是第一次请求这两个页，则两个页都要进行编译。

③ 包含更新的内容的母版页合并到内容页的控件树中。

④ 各个 Content 控件的内容合并到母版页中相应的 ContentPlaceHolder 控件中。

⑤ 浏览器中呈现得到的合并页。

2. 站点导航

可以使用 ASP.NET 站点导航功能为用户导航站点提供一致的方法。随着站点内容的增加以及用户在站点内来回移动网页，管理所有的链接会变得比较困难。ASP.NET 站点导航能够将指向所有页面的链接存储在一个中央位置，并在列表中呈现这些链接，或用一个特定 Web 服务器控件在每页上呈现导航菜单。

一个站点导航主要包括两方面的内容，即站点地图和站点导航控件。

（1）站点地图

站点地图是描述站点逻辑结构的 XML 文件。在网站添加和删除页面时，开发人员只需要更改站点地图文件就可以管理页导航，而不需要修改各个页面本身的导航链接。

创建站点地图最简单的方法是创建一个名为 Web. sitemap 的 XML 文件，该文件按站点的分层形式组织页面。ASP.NET 的默认站点地图提供程序自动选取此站点地图。

尽管 Web. sitemap 文件可以引用其他站点地图提供程序或其他目录中的其他站点地图文件以及同一应用程序中的其他站点地图文件，但该文件必须位于应用程序的根目录中。

在 Web. sitemap 文件中，为网站中的每一页添加一个 siteMapNode 元素。然后，可以通过嵌入 siteMapNode 元素创建层次结构。Web. sitemap 文件结构如下所示。

```
<siteMap>
  <siteMapNode title = "Home" description = "Home" url = "~/default.aspx">
    <siteMapNode title = "Products" description = "Our products"   url = "~/Products.aspx">
        < siteMapNode title = " Hardware" description = " Hardware choices"   url = " ~/
Hardware.aspx" />
        < siteMapNode title = " Software" description = " Software choices"   url = " ~/
Software.aspx" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

在上例中，“硬件”和“软件”页是“产品”siteMapNode 元素的子元素。title 属性定义通常用作链接的文本，description 属性同时用作文档和 SiteMapPath 控件中的工具提示。

（2）站点导航控件

创建一个反映站点结构的站点地图只完成了 ASP.NET 站点导航系统的一部分。导航系统的另一部分是在 ASP.NET 网页中显示导航结构，这样用户就可以在站点内轻松地移动。通过使用下列 ASP.NET 站点导航控件，可以轻松地在页面中建立导航信息。

① SiteMapPath 控件。此控件显示导航路径向用户显示当前页面的位置，并以链接的形式显示返回主页的路径。此控件提供了许多可供自定义链接的外观的选项。

② TreeView 控件。此控件显示一个树状结构或菜单,让用户可以遍历访问站点中的不同页面。单击包含子节点的节点可将其展开或折叠。

③ Menu 控件。此控件显示一个可展开的菜单,让用户可以遍历访问站点中的不同页面。将光标悬停在菜单上时,将展开包含子节点的节点。

可以使用 SiteMapPath 控件创建站点导航,既不用编写代码,也不用显式绑定数据。此控件可自动读取和呈现站点地图信息。但是,如果需要,也可以使用代码自定义 SiteMapPath 控件。

SiteMapPath 控件使用户能够从当前页导航回站点层次结构中较高的页。但是,SiteMapPath 控件不允许从当前页向前导航到层次结构中较深的其他页面。在新闻组或留言板应用程序中,当用户想要查看他们正在浏览的文章的路径时,就可以使用 SiteMapPath 控件。

TreeView 或 Menu 控件能够让用户可以打开节点并直接导航到特定的页。这些控件不会像 SiteMapPath 控件那样直接读取站点地图。相反,用户需要在页上添加一个可读取站点地图的 SiteMapDataSource 控件。然后,将 TreeView 或 Menu 控件绑定到 SiteMapDataSource 控件,从而将站点地图呈现在该页上。

3. 主题和控件外观

(1) 站点导航控件

主题是属性设置的集合,使用这些设置可以定义页面和控件的外观,然后在整个 Web 应用程序中的所有页中一致地应用此外观。

主题由一组元素组成:外观、级联样式表(CSS)、图像和其他资源。主题将至少包含外观。主题是在网站或 Web 服务器上的特殊目录中定义的。

(2) 外观

外观文件具有文件扩展名.skin,它包含各个控件(例如,Button、Label、TextBox 或 Calendar 控件)的属性设置。控件外观设置类似于控件标记本身,但只包含要作为主题的一部分来设置的属性。例如,下面是 Button 控件的控件外观:

```
<asp:button runat = "server" BackColor = "lightblue" ForeColor = "black" />
```

在 theme 文件夹中创建.skin 文件。一个.skin 文件可以包含一个或多个控件类型的一个或多个控件外观。可以为每个控件在单独的文件中定义外观,也可以在一个文件中定义所有主题的外观。

有两种类型的控件外观 :默认外观和已命名外观。

当向页应用主题时,默认外观自动应用于同一类型的所有控件。如果控件外观没有 SkinID 属性,则是默认外观。例如,如果为 Calendar 控件创建一个默认外观,则该控件外观适用于使用本主题的页面上的所有 Calendar 控件。默认外观严格按控件类型来匹配,因此 Button 控件外观适用于所有 Button 控件,但不适用于 LinkButton 控件或从 Button 对象派生的控件。

已命名外观是设置了 SkinID 属性的控件外观。已命名外观不会自动按类型应用于控件,而应当通过设置控件的 SkinID 属性将已命名外观显式应用于控件。通过创建已命名外

观,可以为应用程序中同一控件的不同实例设置不同的外观。

(3) 主题的应用范围

可以定义单个 Web 应用程序的主题,也可以定义供 Web 服务器上的所有应用程序使用的全局主题。定义主题之后,可以使用@ Page 指令的 Theme 或 StyleSheetTheme 属性将该主题放置在单个页上,或者可以通过设置应用程序配置文件中的<pages>元素将其应用于应用程序中的所有页。如果在 Machine.config 文件中定义了<pages>元素,则主题将应用于服务器上的 Web 应用程序中的所有页。

(4) 页面主题

页面主题是一个主题文件夹,其中包含控件外观、样式表、图形文件和其他资源,该文件夹是作为网站中的\App_Themes 文件夹的子文件夹创建的。每个主题都是 \App_Themes 文件夹的一个不同的子文件夹。

(5) 全局主题

全局主题是可以应用于服务器上的所有网站的主题。当维护同一个服务器上的多个网站时,可以使用全局主题定义域的整体外观。

全局主题与页面主题类似,因为它们都包括属性设置、样式表设置和图形。但是,全局主题存储在 Web 服务器的名为\Themes 的全局文件夹中。服务器上的任何网站以及任何网站中的任何页面都可以引用全局主题。

2.4 项目实施

本项目通过 4 个任务介绍创建风格统一的网上书店 Web 网站的过程。

2.4.1 任务 2-1 创建网上书店母版页

1. 任务介绍

网上书店都含有一个用户管理平台,用户可以管理自己的用户信息、地址信息、订单信息等。例如大型的购物网站“京东商城”就有类似的平台。本任务将为网上书店设计一套风格统一的页面,并且希望以后修改界面效果时,能高效、方便地实施。

2. 任务分析

在用 PowerPoint 制作幻灯片时,为使每张幻灯片都有一样的背景、字体、格式等,可以先制作一张母版,之后在幻灯片中运用这个母版,幻灯片的整体风格也就一致了。

同样,如果要高效率地设计拥有统一界面的网页,则可以通过设计母版页来实现。把母版页想象为“网页的母版”,开发人员只需修改这个母版页,就可以使所有的网页拥有一致的界面。

(1) 创建网上书店母版页文件

在现有网站 NetBook 上通过右击添加一个新项,在“添加新项”窗口中,选择“母版页”,如图 2-2 所示。



图 2-2　新建母版页

打开 MasterPage.master，里面有一个 ContentPlaceHolder 控件，如图 2-3 所示。

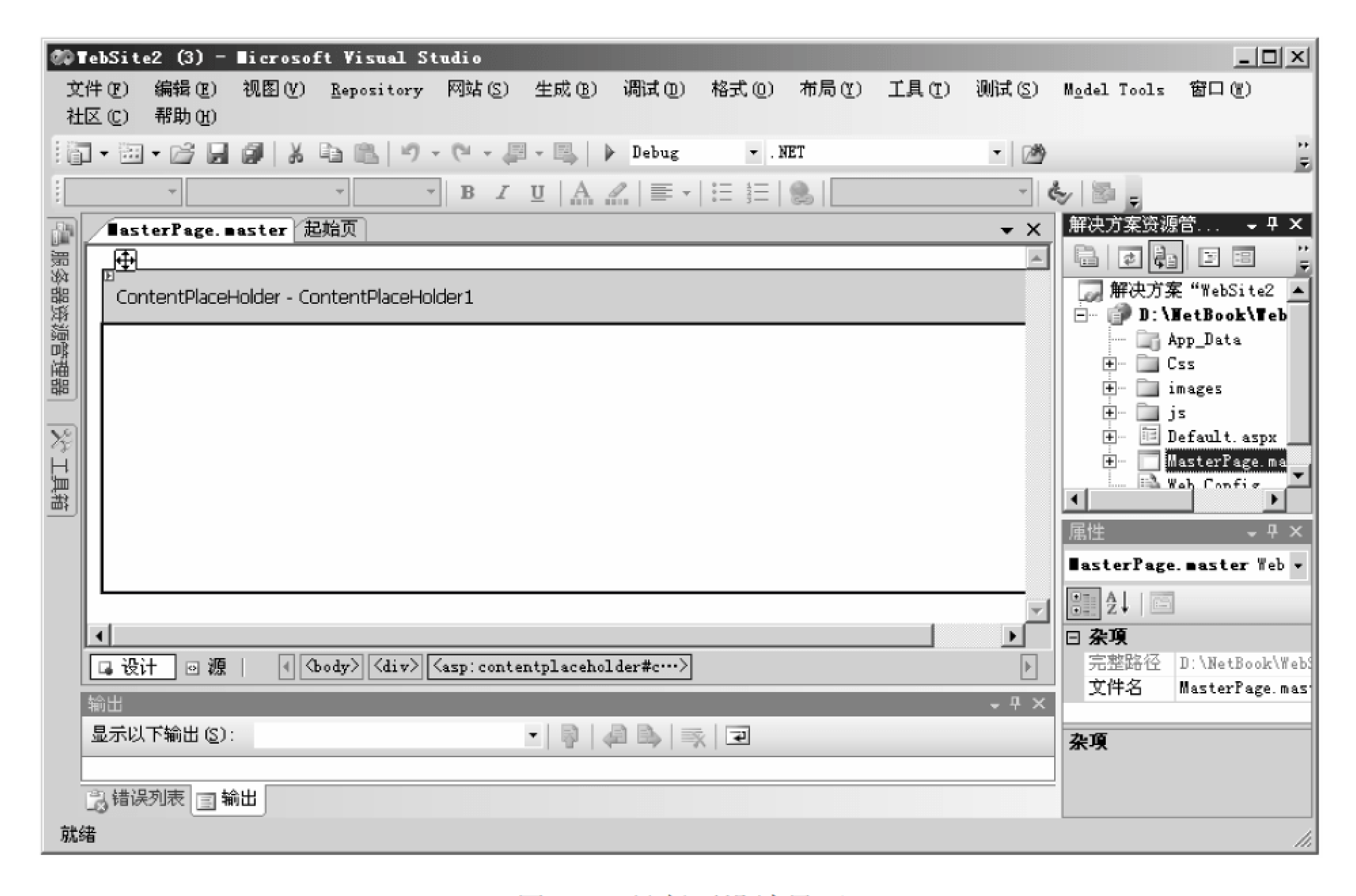


图 2-3　母版页设计界面

由于默认的母版页无法满足要求，需重新设计。首先将母版页上的内容清空，然后设计一个漂亮的界面，最后在页面的右下部分放置一个 ContentPlaceHolder 控件，这样一张简单的母版页就设计好了，可以用它作为其他页面的模板了，如图 2-4 所示。

(2) 创建使用母版页的内容页

在现有网站 NetBook 通过右击添加一个新项，在"添加新项"窗口中，选择"Web 窗体"选项，并且选中"选择母版页"复选框，如图 2-5 所示。

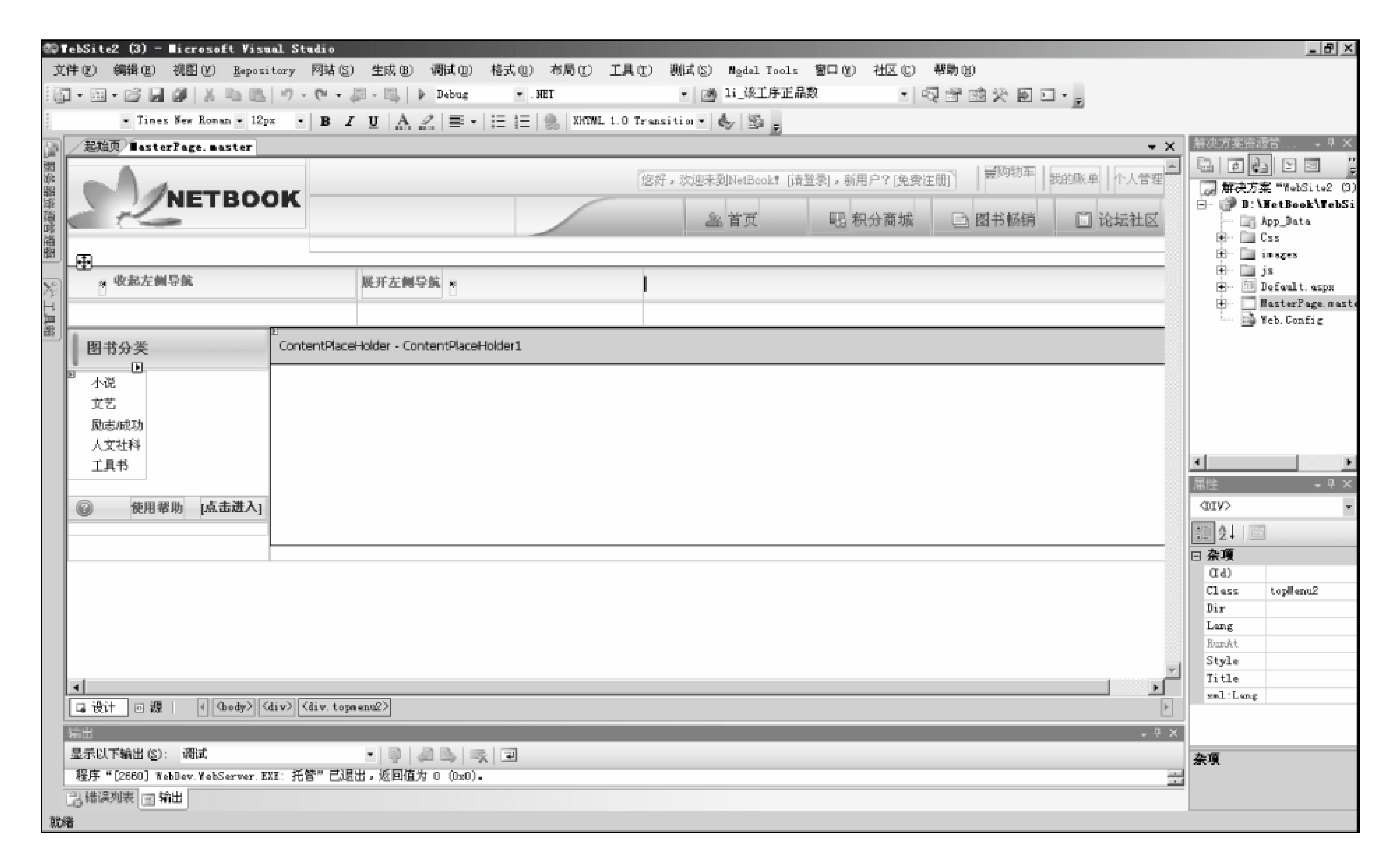


图 2-4　设计好的母版页界面



图 2-5　新建内容页的窗口

选择相应的母版页，如图 2-6 所示。

合成母版页的内容页面与普通页面的源代码相比发生了一些变化，如下所示：

```
<%@ Page Language = "C#" MasterPageFile = "~/MasterPage.master" AutoEventWireup = "true"
CodeFile = "Default2.aspx.cs" Inherits = "Default2" Title = "Untitled Page" %>
<asp:Content ID = "Content1" ContentPlaceHolderID = "ContentPlaceHolder1" Runat = "Server">
</asp:Content>
```

可以看出 Page 指令中多了一个 MasterPageFile="~/MasterPage.master"，这正是页面使用母版页的属性。而一般页面中的＜HTML＞、＜TITLE＞、＜BODY＞、＜FORM＞



图 2-6　选择母版页的窗口

等标记没有了，取而代之的是<asp:Content>服务器控件标记。

切换到设计视图，可以看到一个 Content 控件，它对应母版页的 ContentPlaceHolder1 控件，其中页头和页脚的图片都是灰色的，不能修改，只能在 Content 中进行编辑，如图 2-7 所示。

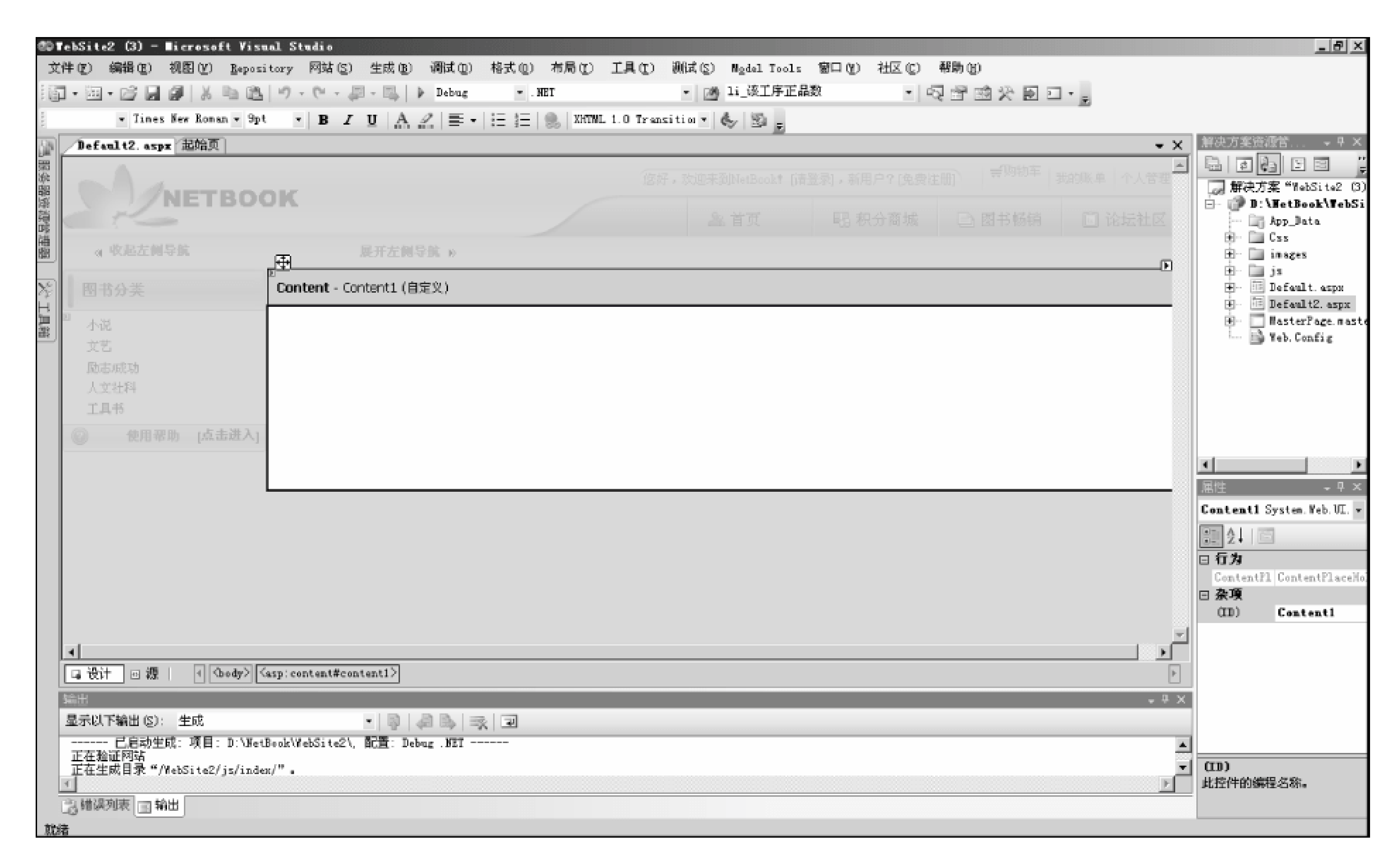


图 2-7　内容页设计界面

在 Content 控件中输入文字“欢迎进入系统”，保存后按 F5 键访问这个页面，效果如图 2-8 所示。

3. 任务完成总结

从运行结果中可以看到虽然只在内容页中加入一些简单的文字，但却能看到带有头尾图片效果的界面，这正是使用母版页的效果。因此，只要在设计页面时都使用相同的母版页，除了 Content 控件可以根据具体的页面做相应设计外，其他的部分都将沿用母版页设计

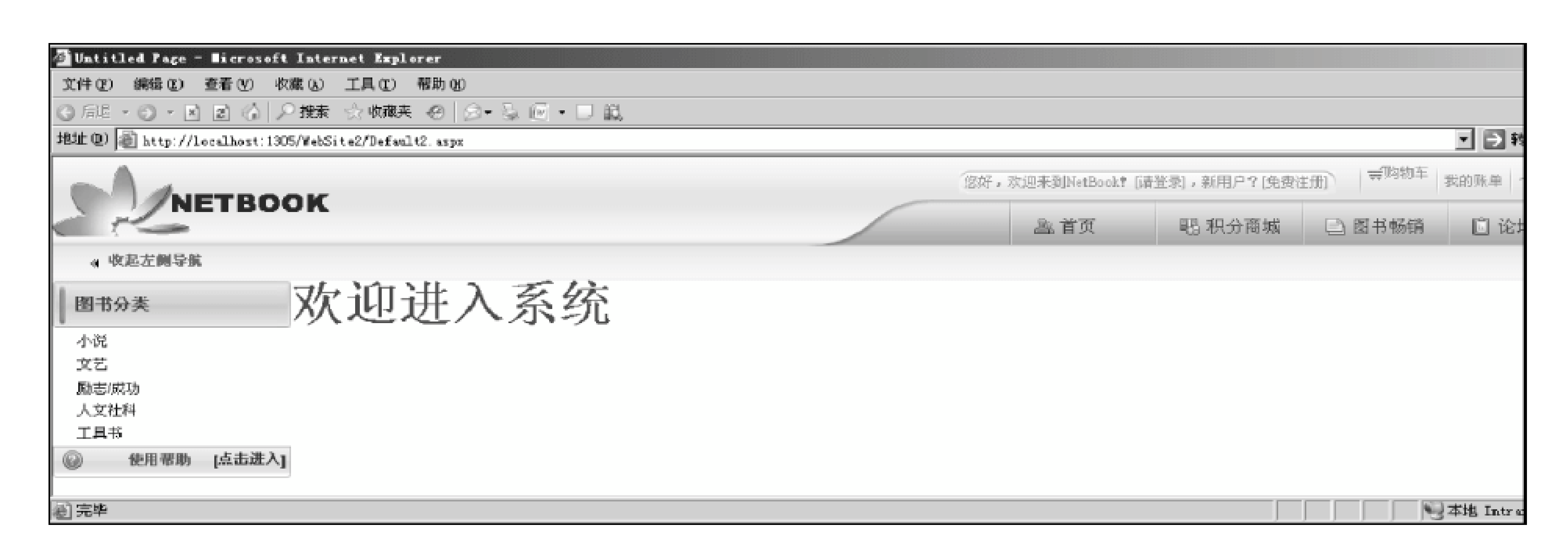


图 2-8　内容运行效果图

好的内容，这样制作一套具有统一风格的网页就不再是难事了。

4. 课题训练与拓展

依据本任务中的步骤设计一个美观的母版页，利用这个母版页生成多张内容页，查看效果。完成后试着修改母版页中局部界面，观察这些内容页是否也发生了统一变化。

2.4.2　任务 2-2　设计网上书店站点地图

1. 任务介绍

在前面的任务中用母版建立了一个具有统一风格的网站，但随着网站网页的增多，新的问题也出现了。开发人员发现网页链接增多后，想回到上一层或更上一层网页的操作变困难了。因此，希望建立一套向导机制，能够清楚地知道目前在网站的位置，并且提供链接方便用户访问其他网页。本任务将设计一套网页导航功能，要求向导功能能快速地部署到目前的网站中并且易于以后的维护和修改。

2. 任务分析

这个任务可以通过使用站点地图来实现。首先来看一下 NetBook 网站目前的结构，如图 2-9 所示。

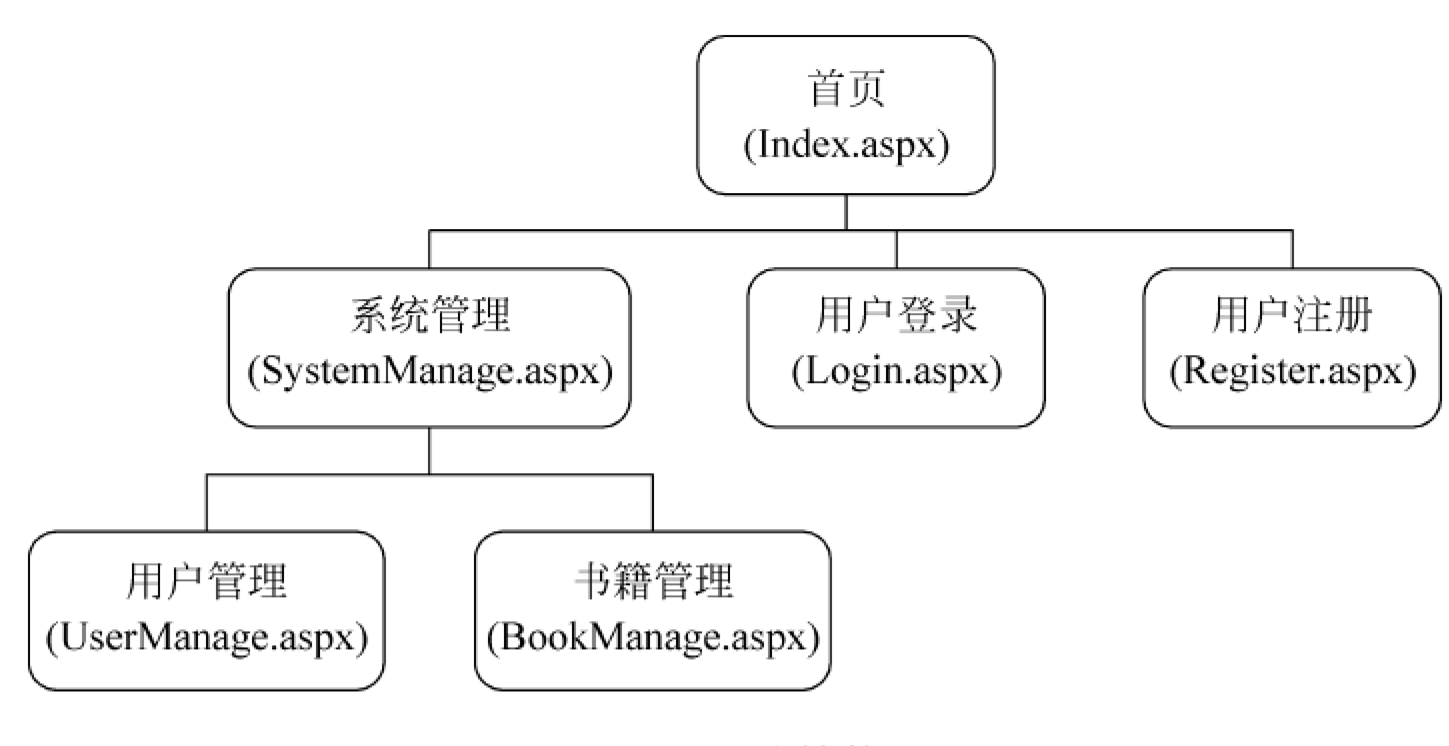


图 2-9　网站结构图

从图 2-9 中可以看到，目前网站定义了三层结构，任务的目标是在访问用户管理页面(UserManage.aspx)时，网页上能提供“首页→系统管理→书籍管理”这种类型的导航功能，这样只要单击导航栏里的“系统管理”，就能转到系统管理(SystemManage.aspx)页面了。

(1) 创建站点地图文件

在现有网站 NetBook 上通过右击添加一个新项，在“添加新项”窗口中，选择“站点地图”，如图 2-10 所示。



图 2-10 新建站点地图的窗口

根据上面分析时绘制的结构图，将其映射成站点地图文件的代码加入到新建的 Web.sitemap 中，如下所示：

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<siteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap - File - 1.0" >
    <siteMapNode url = "Index.aspx" title = "首页"  description = "首页">
        <siteMapNode url = "Login.aspx"  title = "用户登录"  description = "用户登录" />
        <siteMapNode url = "Register.aspx"  title = "用户注册"  description = "用户注册" />
            <siteMapNode url = "SystemManage.aspx"  title = "系统管理"  description = "系统管理">
            <siteMapNode url = "UserManage.aspx"  title = "用户管理"  description = "用户管理" />
            <sitemapNode url = "BookManage.aspx"  title = "书籍管理"  description = "书籍管理" />
        </siteMapNode>
    </siteMapNode>
</siteMap>
```

站点地图中只有两种 XML 节点类型，分别是 siteMap 和 siteMapNode 节点。siteMap 节点是站点地图的根节点，一个站点地图中它是唯一的。siteMapNode 节点是具体描述网站导航页面的节点，它可以多层嵌套。

siteMapNode 节点有三个主要属性，分别是 url 属性、title 属性、description 属性。其中，url 属性记录的是该节点所对应的页面地址，它既可以是网站内部的相对路径，也可以是公网上的地址链接。title 属性描述了节点的显示名称。description 属性记录了该页面的具体描述信息，当鼠标悬停在该链接上时，显示的即为 description 记录的信息。

打开已有的 BookManage.aspx 文件，在 Content 控件内加入一个 SiteMapPath 控件，控件会按之前在 Web.sitemap 文件中写的结构自动创建链接路径，如图 2-11 所示。

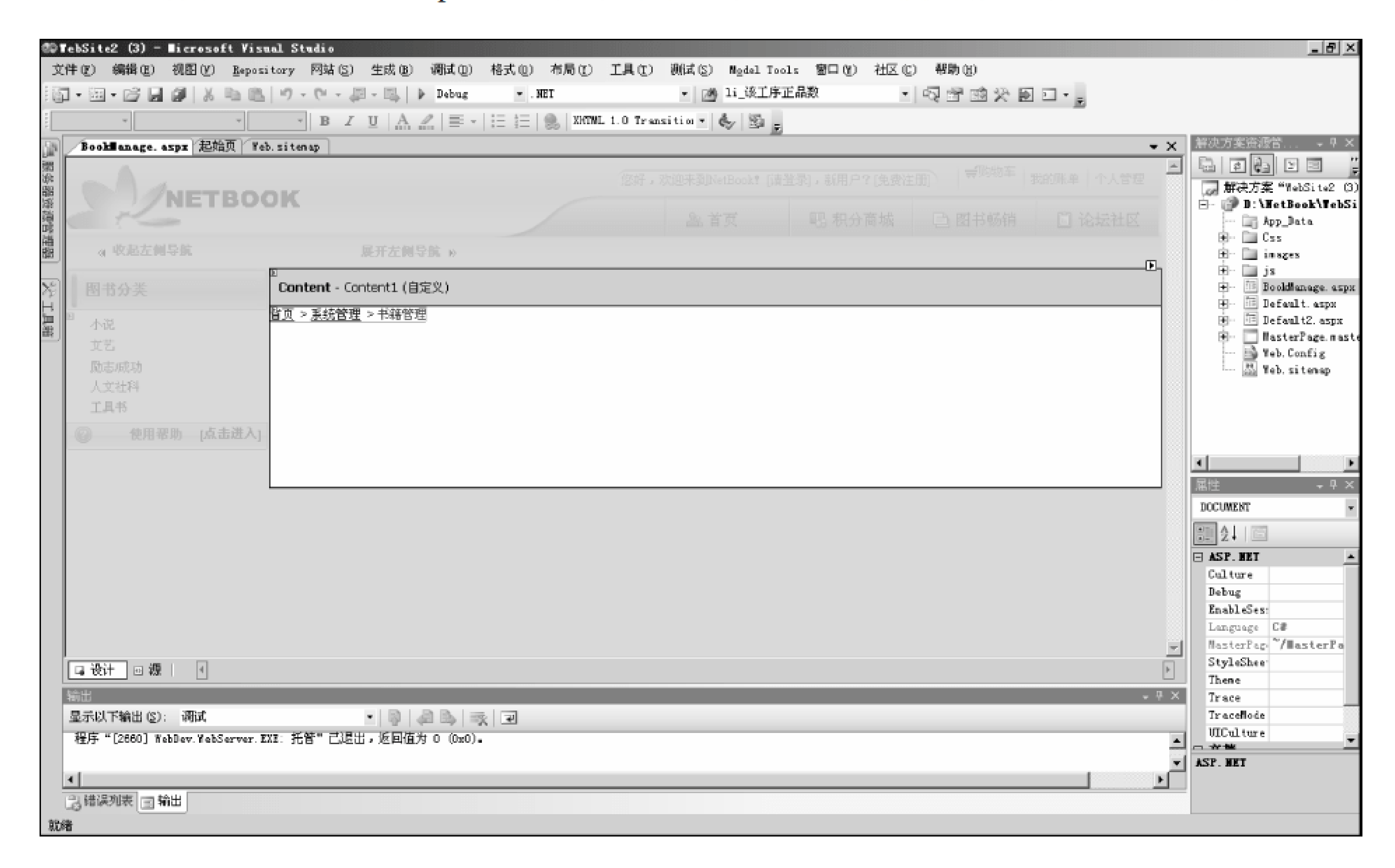


图 2-11　加入 SiteMapPath 控件的页面

运行 BookManage.aspx，效果如图 2-12 所示。



图 2-12　具有站点导航栏的运行界面

(2) 将站点地图文件添加到母版页

在网页中加入站点地图成功后，会发现在每张网页中加 SiteMapPath 控件还是太麻烦。可以在母版页中使用 SiteMapPath 控件，这样不但只需加一次控件，而且使用母版的页面都拥有相同风格的导航栏，如图 2-13 所示。通过运行结果可以看到所有使用母版的页面都加上了导航栏。

3. 任务完成总结

本任务主要介绍网站地图的设计过程。网站地图设计使用 SiteMapPath 控件读取 Web.sitemap 文件，网站地图链接信息时使用 Web.sitemap 文件进行配置。

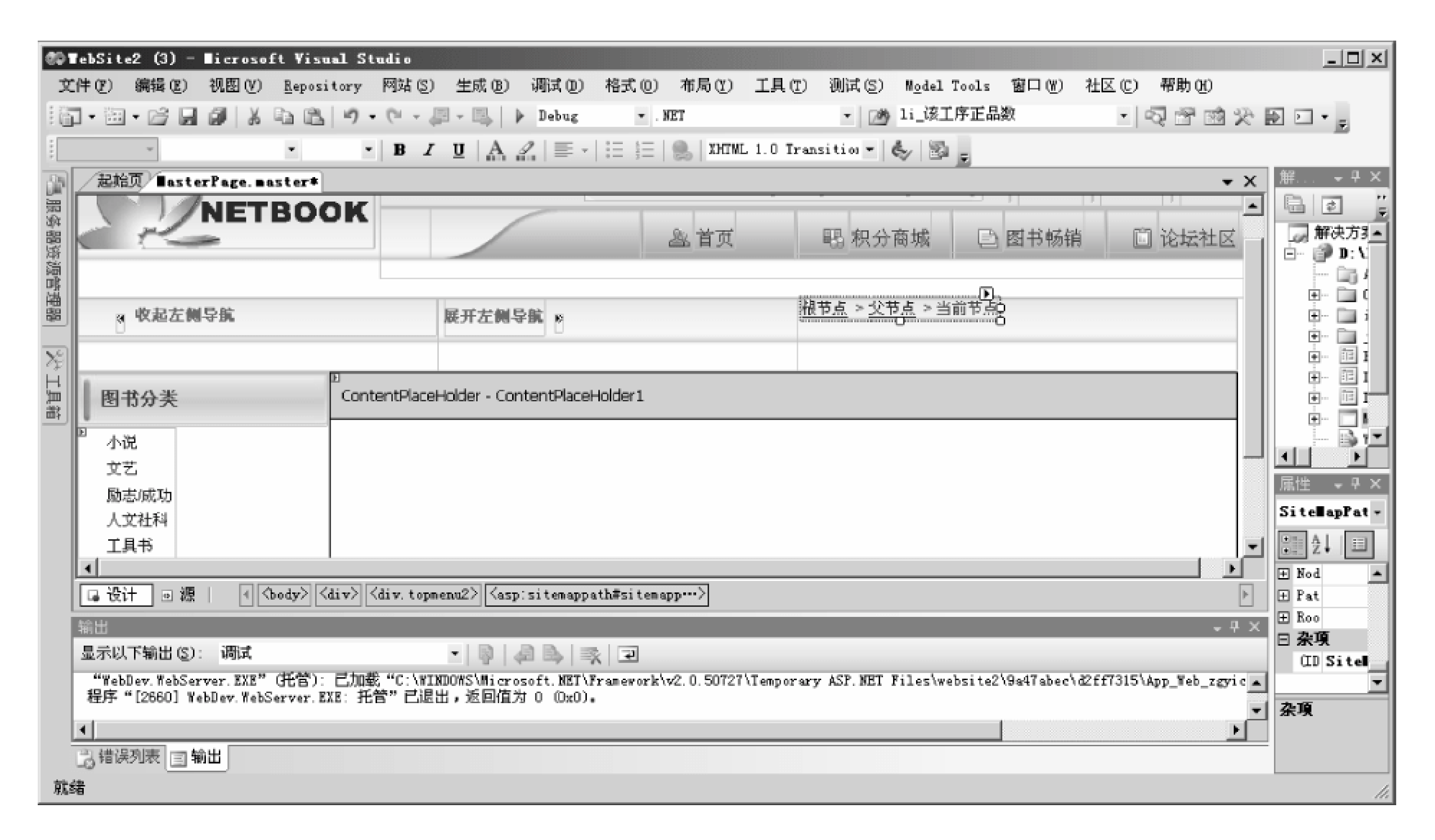


图 2-13 加入 SiteMapPath 控件的母版页

4. 课堂训练与拓展

依据本任务中的步骤为网上书店设计一个合理的站点导航栏。

2.4.3 任务 2-3 设计网上书店皮肤文件

1. 任务介绍

前面学习的母版可以用于网页整体风格的设计,但在细节上母版却起不了作用。比如设计的多个网页中都用到了保存功能的图片按钮,按钮上希望使用相同的图片,并且希望按钮的大小、颜色看起来都一样。最简单的办法当然是在设计这些相同样式的按钮时,开发人员为它们设置相同的属性,但这种方式必须一个一个设置,而且一旦设计好后需要修改效果时,也必须一个一个修改,这种设置方法不灵活。可以使用皮肤文件来解决这个问题,提高设计的效率。本任务将使用皮肤设计技术设置具有相同样式控件的外观,并且易于以后的维护与修改。

2. 任务分析

我们可以在主题目录中创建 .skin 文件。在这个 skin 文件中为各种控件设计一个或多个皮肤,然后对页面中的控件设置 SkinID 属性,将指定的皮肤应用于控件。

(1) 创建网上书店皮肤文件

在现有网站 NetBook 上通过右击添加一个新项,在"添加新项"窗口中,选择"外观文件",如图 2-14 所示。

单击"添加"按钮后,系统会弹出一个消息框,询问是否将文件放到 App_Themes 文件夹中,如图 2-15 所示。



图 2-14　新建皮肤文件的窗口

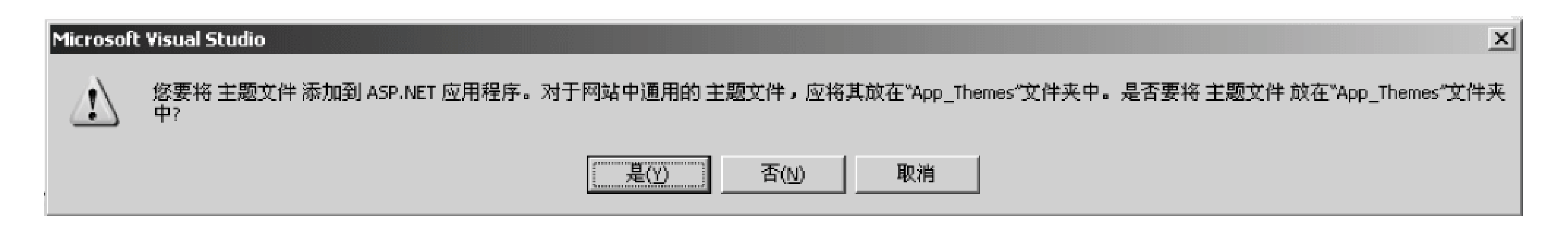


图 2-15　新建皮肤文件的提示窗口

单击"是"按钮，将在站点中创建一个 App_Themes 文件夹，并在该文件夹下生成一个 SkinFile 子文件夹，里面有一个新的皮肤文件 SkinFile.skin，如图 2-16 所示。

打开 SkinFile.skin 文件，在里面添加一个图片按钮的样式，代码如下：

```
<asp:ImageButton  runat = "server"  SkinId = "DeleteImageButtonSkin" ImageUrl = "~/images/m_an_
del.gif"  />
```

打开 Web.Config 文件，在<system.web>标记中添加一段配置代码，如下：

```
<pages theme = "SkinFile" />
```

(2) 创建使用皮肤的页面

打开一个网页，在网页中加入一个 ImageButton 控件，设置它的 SkinID 属性为 DeleteImageButtonSkin，如图 2-17 所示。

运行网页，网页中 ImageButton 控件的样式和在皮肤文件中定义的 ImageButton 的样式完全一样，如图 2-18 所示。

3. 任务完成总结

通过运行结果，可以看到皮肤文件就是定义一些控件的公共样式，以此来达到一次设计多次使用的目的，另外也可以用于保证控件的风格一致。



图 2-16　新建的 App_Themes 文件夹

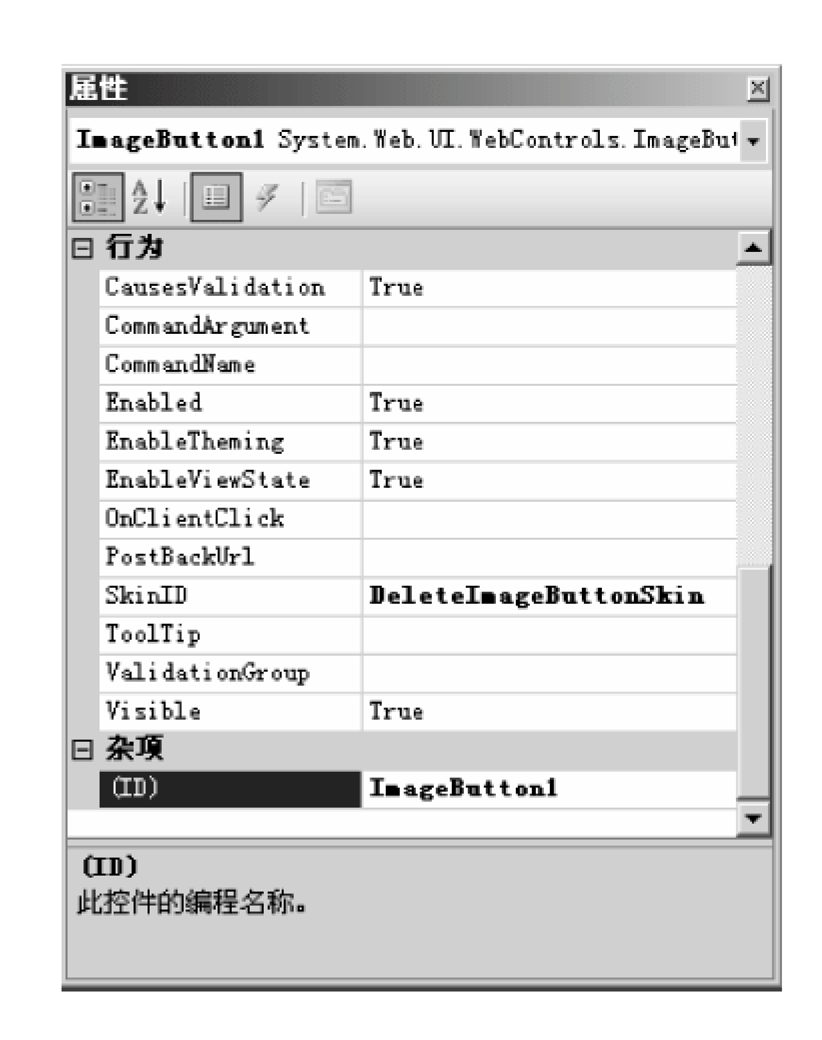


图 2-17　ImageButton 控件属性窗口



图 2-18　使用了皮肤的运行界面

4. 课堂训练与拓展

依据本任务中的步骤设计一套风格一致的控件。

2.4.4　任务 2-4　使用 CSS 文件统一网上书店主题样式

1. 任务介绍

在设计一些报表网页时，比如显示人员信息、用户信息等，由于工作量比较大，会由多个开发人员设计，结果设计出来的网页显示效果五花八门。此外有的开发人员美工水平比较差，做的网页不美观，如图 2-19 所示。因此希望能重新设计这些页面，具有统一的、美观的表格风格。本任务将设计具有统一风格的表格。

2. 任务分析

CSS 是 Cascading Style Sheet 的缩写，译作“级联样式表”，是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。可以使用 CSS 设计统一样式的表格。



图 2-19　不美观的表格网页

（1）创建 CSS 文件

在现有网站 NetBook 上通过右击添加一个新项，在“添加新项”窗口中，选择级联样式表文件，如图 2-20 所示。



图 2-20　新建级联样式表文件的窗口

将样式代码加入到新建的 StyleSheet.css 中，如下所示：

```
.table-TitleBG {
    font-weight: bold;
    font-size: 12px;
    line-height: 150%;
    background-color: #f7faff;
    text-decoration: none
    }

.table-Kang {
    background-color: #91afdd;
    border: 1px solid #0066cc
    }
```

```
.table-Sbg {
    font-size: 12px;
    line-height: 150%;
    background-color: #ecf4ff
    }
```

(2) 创建使用 CSS 的页面

在页面中加入样式引用,如下所示:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Default4.aspx.cs" Inherits="Default4" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<link rel="stylesheet" href="StyleSheet.css" type="text/css" />
<table  border="0" cellpadding="4" cellspacing="1" class="table-Kang" style="width:
518px; height: 78px;">
<tr>
<td class="table-TitleBG">用户名</td>
<td class="table-Sbg" width=15% style="text-align: left">张三</td>
<td class="table-TitleBG">出生年月</td>
<td class="table-Sbg" width=15% style="text-align: left">1970-1-1</td>
<td class="table-TitleBG">性别</td>
<td class="table-Sbg" width=10% style="text-align: left">男</td>
</tr>
<tr>
<td class="table-TitleBG">电话</td>
<td class="table-Sbg" style="text-align: left">88888888</td>
<td class="table-TitleBG">E-mail</td>
<td class="table-Sbg" colspan="3" style="text-align: left">zhang@netbook.com </td>
</tr>
<tr>
<td class="table-TitleBG">地址</td>
<td class="table-Sbg" colspan="5" style="text-align: left">常州信息职业技术学院计算
机系</td>
</tr>
</table>
</asp:Content>
```

运行网页,网页中表格的效果发生了明显变化,如图 2-21 所示。



图 2-21　使用了级联样式表的运行界面

3. 任务完成总结

通过运行结果，我们可以看到如果多个网页使用级联样式表文件定义好的样式，可以使网页具有相同的界面效果。如果需要修改界面效果，也只需要修改级联样式表文件，所有使用这些样式的网页都会相应改变。

4. 课堂训练与拓展

依据本任务中的步骤设计一个使用级联样式表的图书页面。

2.5　项目总结

本项目结合实际任务介绍了常用的设计统一风格网页的 ASP.NET 新技术：母版页、站点地图、皮肤、级联样式表等。由于任务要求的限制，还有一些复杂用法没有介绍，比如母版页的嵌套、母版页的事件处理、站点导航控件的自定义使用等。读者可以在掌握了基本知识后深入学习。

2.6　项目实训

任务描述

表 2-1 是一个旅游网站的栏目设计，要求按栏目设计一个具有统一风格的旅游网站。

表 2-1　某旅游网站栏目内容

栏　目	服务分类	栏　目	服务分类
旅游动态	国外旅游信息	便民服务	问题解答
	国内旅游信息		出入境办理
	精品游记推荐		消费者维权
风情	文物古迹	旅游线路	精品线路
	风景导航		旅游公司介绍
	节日活动	旅游一线通	天气介绍
	历史文化		宾馆介绍
	民俗风情		交通路线介绍
	特别推荐		购物介绍
网上预订	网上预订酒店		旅游地图
	网上火车票预订		餐饮介绍
	网上汽车票预订		导游信息
	网上飞机票预订		旅游租车
	网上咨询	旅游反馈	景点评价
			旅游投诉

创建能够与用户交互的网站

3.1 项 目 介 绍

通过前面的项目,我们已经能够设计出风格一致、界面美观的网站页面。但是静态的网站只能让用户访问到设计好的内容,却无法获取用户的反馈信息。比如,为了更好地为用户服务,通常希望用户能把他们的姓名、电话、邮箱等信息告知客服部门。但是,由于用户都是通过网络访问网站,无法通过传统的纸质填表方式获取信息,因此希望能够设计出与用户交互的页面,方便用户与网站进行交互。

3.2 项 目 分 析

Web 应用程序网站通过网络方式收集用户信息,这就需要设计出能和用户交互的页面。用户可以通过这些页面输入自己的个人信息,并提交到服务器上。ASP.NET 中使用服务器控件设计与用户交互的页面。本项目中将设计一个用户注册界面,通过用户注册界面的设计来讨论交互性 Web 网站的设计过程。

用户名、密码、真实姓名、身份证、出生日期、联系方法等信息可以使用文本框输入。性别只能为男或女,可以进行唯一性选择。注册用户所在省份的分布信息可以使用下拉列表选择。如果有特殊要求的网站需要用户输入照片,一般可以使用上传控件完成照片上传。

输入的用户信息一般需要进行有效性验证。为防止无效的空数据进入数据库,输入的所有内容需要进行非空验证。为防止密码输入出错,密码需要进行两次比对验证。出生日期、身份证号码等数据都有特殊的格式要求,需要进行正则验证。已经被其他用户使用过的用户名不可以重复使用,需要进行数据比对判断验证。

上述讨论的创建用户注册页面的内容,都是通过使用服务器控件来完成的。在实施本项目前,先学习一些设计交互式页面必须的控件。

3.3 相 关 知 识

1. HTML 服务器控件

对服务器公开的 HTML 元素,可对其进行编程。HTML 服务器控件公开一个对象模

型，该模型十分紧密地映射到相应控件所呈现的 HTML 元素。

HTML 服务器控件属于 HTML 元素(或采用其他支持的标记的元素，例如 XHTML)，它包含多种属性，使其可以在服务器代码中进行编程。默认情况下，服务器上无法使用 ASP.NET 网页中的 HTML 元素。这些元素将被视为不透明文本并传递给浏览器。但是，通过将 HTML 元素转换为 HTML 服务器控件，可将其公开为可在服务器上编程的元素。

HTML 服务器控件的对象模型紧密映射到相应元素的对象模型。例如，HTML 属性在 HTML 服务器控件中作为属性公开。

页中的任何 HTML 元素都可以通过添加属性 runat＝"server"转换为 HTML 服务器控件。在分析过程中，ASP.NET 页框架将创建包含 runat＝"server"属性的所有元素的实例。若要在代码中以成员的形式引用该控件，则还应为该控件分配 id 属性。

页框架为页中的 HTML 元素提供了预定义的 HTML 服务器控件：form 元素、input 元素(文本框、复选框、“提交”按钮)、select 元素，等等。这些预定义的 HTML 服务器控件具有一般控件的基本属性，此外每个控件通常提供自己的属性集和自己的事件。

HTML 服务器控件提供以下功能：

① 可在服务器上使用面向对象技术对其进行编程的对象模型。每个服务器控件都公开一些属性 (Property)，可以使用这些属性 (Property) 在服务器代码中以编程方式来操作该控件的标记属性 (Attribute)。

② 提供一组事件，可以为其编写事件处理程序，方法与在基于客户端的窗体中大致相同，所不同的是事件处理是在服务器代码中完成的。

③ 自动维护控件状态。在页到服务器的往返行程中，将自动对用户在 HTML 服务器控件中输入的值进行维护并发送回浏览器。

④ 与 ASP.NET 验证控件进行交互，因此可以验证用户是否已在控件中输入了适当的信息。

⑤ 数据绑定到一个或多个控件属性。

⑥ 支持样式(如果在支持级联样式表的浏览器中显示 ASP.NET 网页)。

⑦ 直接可用的自定义属性。可以向 HTML 服务器控件添加所需的任何属性，页框架将呈现这些属性而不会更改其任何功能。这允许用户向控件添加浏览器特定属性。

2. Web 服务器控件

Web 服务器控件比 HTML 服务器控件具有更多的内置功能。Web 服务器控件不仅包括窗体控件(例如按钮和文本框)，而且还包括特殊用途的控件(例如日历、菜单和树视图控件)。Web 服务器控件与 HTML 服务器控件相比更为抽象，因为其对象模型不一定反映 HTML 语法。

Web 服务器控件是设计侧重点不同的另一组控件。它们不必一对一地映射到 HTML 服务器控件，而是定义为抽象控件。在抽象控件中，控件所呈现的实际标记与编程所使用的模型可能截然不同。例如，RadioButtonList Web 服务器控件可以在表中呈现，也可以作为带有其他标记的内联文本呈现。

Web 服务器控件包括传统的窗体控件，例如按钮、文本框和表等复杂控件。它们还包括提供常用窗体功能(例如在网格中显示数据、选择日期、显示菜单等)的控件。

Web 服务器控件除了提供 HTML 服务器控件的上述所有功能(不包括与元素的一对一映射)外,还提供以下附加功能:

① 功能丰富的对象模型,该模型具有类型安全编程功能。

② 自动浏览器检测,控件可以检测浏览器的功能并呈现适当的标记。

③ 对于某些控件,可以使用 Templates 定义自己的控件布局。

④ 对于某些控件,可以指定控件的事件是立即发送到服务器,还是先缓存然后在提交该页时引发。

⑤ 支持主题,可以使用主题为站点中的控件定义一致的外观。

⑥ 可将事件从嵌套控件(例如表中的按钮)传递到容器控件。

下面介绍一些常用的标准 Web 服务器控件,使用这些控件设计具有交互性的 Web 网站非常方便,可以极大地减少代码量。

(1) Label 控件

Label 控件为用户提供了一种以编程方式设置 ASP.NET 网页中文本的方法,可以在程序运行时通过代码更改页面中的文本内容。

Label 控件最常用的属性就是 Text 属性,用于设置在标签控件中显示的文本。例如当用户单击按钮提交了信息以后,如果后台服务器成功保存了信息,可以编写下面的代码,提示用户操作成功。设置 Label 控件的 ID 属性为 Message,在页面显示提示信息的代码如下:

```
Label_Message.Text = "信息已成功保存";
```

(2) TextBox 控件

TextBox 控件为用户提供了一种向 Web 窗体页中输入信息(包括文本、数字和日期)的方法。在默认情况下,只能使 TextBox 控件输入单行文本,如果需要输入比较特殊的信息,如密码或内容比较多的多行信息时,可以利用"TextMode"属性来配置 TextBox Web 服务器控件类型,如表 3-1 所示。

表 3-1 TextMode 属性

配 置	说 明	样 式
SingleLine	在默认情况下,只能使 TextBox 控件输入单行信息。此外通过设置控件的 MaxLength 属性还可以限制控件接受的字符数	张三
Password	与单行 TextBox 控件类似,但用户输入的字符将用星号(*)屏蔽,以隐藏这些信息。使用为密码设置的 TextBox 控件有助于确保其他人员观察用户输入密码时无法确知该密码。但是,输入的密码文本没有以任何方式进行加密,应像保护任何其他机密数据那样对它进行保护。例如,为了获得最大限度的安全性,在发送其中带有密码的窗体时,可以使用安全套接字层(SSL)和加密	••••••
MultiLine	用户在显示多行并允许文本换行的框中输入信息。此外通过设置控件的 Width/Height 属性值或 Columns/Row 属性值以确定控件显示的宽度和行数	张三任软件组组长 李四任软件组组员 王五任软件组组员

TextBox 控件最常用的属性就是 Text 属性，用于获取或者设置文本框中的文本。例如，获取 TextBox_UserName 中的文本，将其保存到字符串变量 userName 中，代码如下：

```
string  userName = TextBox_UserName.Text;
```

TextBox 控件具有一些比较有用的事件。当用户离开 TextBox 控件时，该控件将引发 TextChanged 事件。默认情况下，并不立即引发该事件；而是当提交 Web 窗体页时才在服务器上引发。但可以指定 TextBox 控件在用户离开该字段之后马上将页面提交给服务器。

TextBox 服务器控件并非每当用户输入一个键击就引发事件，而是仅当用户离开该控件时才引发事件。可以让 TextBox 控件引发在客户端脚本中处理的客户端事件，这有助于响应单个键击。

(3) RadioButton 控件

通常将两个或多个单独的 RadioButton 组合在一起，提供一种可以在一组互相排斥的预定义选项中进行选择的方式。可以通过单选按钮的 Text 属性设置它的标题，通过 TextAlign 属性设置标题的对齐方式，通过 Checked 属性用来获取或设置它的选中状态。

此外在单选按钮中有一个很重要的 GroupName 属性，用于分组。单选按钮很少单独使用，通过该属性可以对多个单选按钮进行分组。在一个组内，每次只能选择一个单选按钮。可以用下列方法创建分组的单选按钮：先向页中添加多个 RadioButton Web 服务器控件，然后将所有这些控件手动分配到一个组中，即给这些控件起一个相同的组名。这种情况下，具有相同组名的所有单选按钮视为这个组的组成部分。

例如，在页面中需要填写性别，这时可以添加两个单选按钮，分别将它们的 Text 属性设置为“男”和“女”，再将两个控件的 GroupName 属性都设置为 Sex 即可。

单个 RadioButton 控件在用户单击该控件时引发 CheckedChanged 事件。默认情况下，这一事件并不导致向服务器发送页面，但通过将 AutoPostBack 属性设置为 True，可以使该控件强制立即发送。

无论 RadioButton 控件是否发送到服务器，通常都没有必要为 CheckedChanged 事件创建事件处理程序。相反，更常见的做法是在窗体已被某个控件(如 Button 控件)发送到服务器时测试选定了哪个按钮。

(4) 按钮控件

按钮 Web 服务器控件使用户可以提交已完成的表单或要执行特定的命令。Web 服务器控件包括三种按钮，每种按钮在网页上显示的方式都不同。表 3-2 列出了可以使用 Web 服务器控件创建的按钮的类型。

表 3-2　按钮类型

控　件	说　　明	样　式
Button	显示一个标准命令按钮，该按钮呈现为一个 HTML Input 元素。	Button
LinkButton	呈现为页面中的一个超链接。但是，它包含使窗体被发回服务器的客户端脚本(可以使用 HyperLink Web 服务器控件创建真实的超链接)	LinkButton
ImageButton	允许用户将一个图形指定为按钮。这对于提供丰富的按钮外观非常有用。ImageButton 控件还查明用户在图形中单击的位置，这使用户能够将按钮用作图像映射	

当用户单击这三种类型按钮中的任何一种时，都会向服务器提交一个 Web 窗体。这使得在基于服务器的代码中，网页被处理，任何挂起的事件被引发。这些按钮还可引发它们自己的 Click 事件，可以为这些事件编写事件处理程序。

(5) DropDownList 控件

DropDownList(下拉列表)控件用来在 Web 页面中创建一个下拉列表框，可以单击这个下拉列表框右边的箭头按钮显示列表，然后从中选择一项，注意 DropDownList 控件不支持多重选择模式。DropDownList 控件实际上是列表项的容器，这些列表项都属于 ListItem 类型。每一个 ListItem 对象都是带有自己的属性的单独对象。表 3-3 说明了这些属性。

表 3-3　ListItem 常用属性

属　性	说　　明
Text	指定在列表中显示的文本
Value	包含与某个项相关联的值。设置此属性可使用户将该值与特定的项关联而不显示该值。例如，用户可以将 Text 属性设置为美国某个州的名称，而将 Value 属性设置为该州的邮政区名缩写
Selected	通过一个布尔值指示是否选择了该项

若要处理列表项，可以使用 DropDownList 控件的 Items 集合。Items 集合是一个标准集合，可以向它添加项对象，也可以从中删除项或清除集合等。一般我们在 Items 属性中打开“ListItem 集合编辑器”对话框处理列表项，如图 3-1 所示。

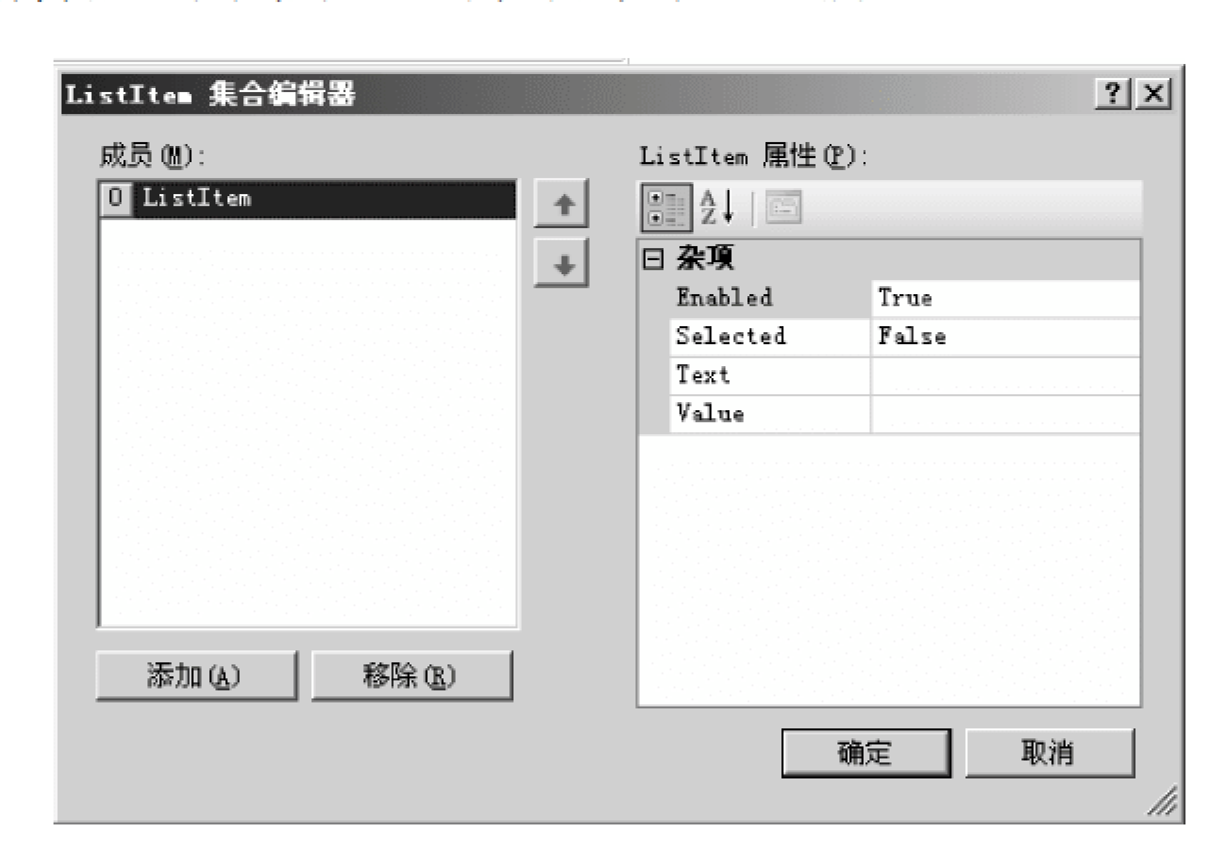


图 3-1　“ListItem 集合编辑器”对话框

在图 3-1 所示对话框中，单击“添加”按钮向列表框中添加一个新项，此时在对话框的右边将显示该项的属性。每个列表项都包含 4 个属性：Enabled(用来设置该项是否可用)、Selected(用来设置该项是否处于选中状态)、Text(用来设置列表项的标题)和 Value(用来设置列表项的值)。对话框中的“移除”按钮用来删除在“成员”列表框中当前选中的项。添加完列表项之后，单击“确定”按钮关闭该对话框并把列表项添加到下拉列表框中。

此外也可以用代码向下拉列表框控件中添加列表项，例如下面的代码：

```
DropDownList1.Items.Add(new ListItem("第一项"));
DropDownList1.Items.Add(new ListItem("第二项"));
```

```
DropDownList1.Items.Add(new ListItem("第三项"));
```

下拉列表框中还包含 SelectedIndex 属性和 SelectedItem 属性，它们分别表示当前选择项的索引和当前选择项。可以在运行时使用代码通过这两个属性来获取下拉列表控件的当前选择项。

当用户选择一项时，DropDownList 控件将引发一个事件（SelectedIndexChanged 事件）。默认情况下，此事件不会导致将页发送到服务器，但可以通过将 AutoPostBack 属性设置为 true 使此控件强制立即发送。

（6）FileUpload 控件

使用 FileUpload 控件，可以为用户提供一种将文件从用户的计算机发送到服务器的方法。该控件在允许用户上传图片、文本文件或其他文件时很有用，控件的样式如图 3-2 所示。



图 3-2　FileUpload 控件

FileUpload 控件显示一个文本框，在此用户可以输入希望上传到服务器的文件名。该控件还显示一个“浏览”按钮，该按钮显示一个文件导航对话框。出于安全方面的考虑，不能将文件名预加载到 FileUpload 控件中。

用户选择要上传的文件并提交页面后，该文件作为请求的一部分上传。文件将被完整地缓存在服务器内存中。文件完成上传后，页代码开始运行。

可以通过下面的 FileUpload 控件属性访问上传的文件：FileBytes 属性中公开的字节数组，FileContent 属性中公开的流，PostedFile 属性中类型 HttpPostedFile 的对象。PostedFile 对象公开某些属性，如 ContentType 和 ContentLength 属性，这些属性提供有关上传文件的信息。

在代码运行时，可以检查文件的特征，例如文件的名称、大小和 MIME 类型，然后可以保存该文件。可以将文件当作字节数组或流来使用。另外，FileUpload 控件和 HttpPostedFile 对象都支持将文件写入磁盘的 SaveAs 方法。

对所上传文件的保存位置，没有固有限制。但是，若要保存文件，ASP.NET 进程必须具有在指定位置创建文件的权限。此外，还可能将应用程序配置为要求使用绝对路径（而不是相对路径）来保存文件，这是一种安全措施。如果将 httpRuntime 元素（ASP.NET 设置架构）配置元素的 requireRootedSaveAsPath 属性设置为 True（默认值），则在保存上传的文件时必须提供绝对路径。

可上传的最大文件的大小取决于 MaxRequestLength 配置设置的值。如果用户试图上传大于最大允许值的文件，则上传会失败。

使用 FileUpload 控件，用户可能上传潜在有害的文件，这包含脚本文件和可执行文件。无法预先限制用户可以上传的文件。如果希望限制用户可以上传的文件的类型，则必须在上传文件后检查文件特征（例如，文件扩展名和文件的 ContentType 属性值）。

（7）TreeView 控件

TreeView Web 服务器控件用于以树形结构显示分层数据，如目录或文件目录。控件的样式如图 3-3 所示。

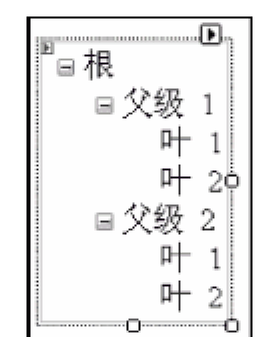


图 3-3　TreeView 控件

它支持以下功能：

① 自动数据绑定，该功能允许将控件的节点绑定到分层数据(如 XML 文档)。

② 通过与 SiteMapDataSource 控件集成提供对站点导航的支持。

③ 可以显示为可选择文本或超链接的节点文本。

④ 可通过主题、用户定义的图像和样式自定义外观。

⑤ 通过编程访问 TreeView 对象模型，可以动态地创建树，填充节点以及设置属性等。

⑥ 可以通过客户端到服务器的回调填充节点(在受支持的浏览器中)。

⑦ 能够在每个节点旁边显示复选框。

TreeView 控件由一个或多个节点构成。树中的每个项都被称为一个节点，由 TreeNode 对象表示。每个 TreeNode 对象包含 4 个 UI 元素。图 3-4 中显示了这些元素，表 3-4 中对其进行了描述。

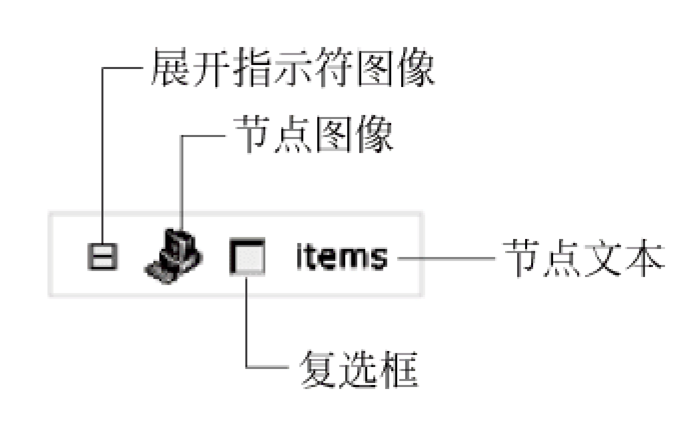


图 3-4 节点呈现

表 3-4 TreeNode 对象包含元素

节点类型	说　明
“展开/折叠”图像	一个可选图像，指示是否可以展开节点以显示子节点。默认情况下，如果节点可以展开，此图像将为加号 [+]，如果此节点可以折叠，则图像为减号 [-]
复选框	复选框是可选的，以允许用户选择特定节点
“节点”图像	可以指定要显示在节点文本旁边的节点图像
“节点”文本	节点文本是在 TreeNode 对象上显示的实际文本。节点文本的作用类似于导航模式中的超链接或选择模式中的按钮

TreeView 控件的节点有 3 种类型，表 3-5 描述了这 3 种不同的节点类型。

表 3-5 TreeView 节点类型

节点类型	说　明
根节点	没有父节点，但具有一个或多个子节点的节点
父节点	具有一个父节点，并且有一个或多个子节点的节点
叶节点	没有子节点的节点

尽管一个典型的树形结构只有一个根节点，但 TreeView 控件允许用户向树形结构中添加多个根节点。

每个节点都具有一个 Text 属性和一个 Value 属性。Text 属性的值显示在 TreeView 控件中，而 Value 属性则用于存储有关该节点的任何附加数据，例如传递给与节点相关联的回发事件的数据。

单击 TreeView 控件的节点时，将引发选择事件(通过回发)或导航至其他页。未设置 NavigateUrl 属性时，单击节点将引发 SelectedNodeChanged 事件，用户可以处理该事件，从而提供自定义的功能。每个节点还都具有 SelectAction 属性，该属性可用于确定单击节点时发生的特定操作，例如展开节点或折叠节点。若要在单击节点时不引发选择事件而导航至其他页，可将节点的 NavigateUrl 属性设置为除空字符串之外的值。

在 TreeView 控件中显示数据有多种方式，最常见的两种分别是静态显示数据和在代

码中绑定数据。

① 静态显示数据。

```
<asp:TreeView ID = "TreeView1" Runat = "server">
  <Nodes>
    <asp:TreeNode Value = "Child1" Expanded = "True" Text = "1">
      <asp:TreeNode Value = "Grandchild1" Text = "A" />
      <asp:TreeNode Value = "Grandchild2" Text = "B" />
    </asp:TreeNode>
    <asp:TreeNode Value = "Child2" Text = "2" />
    <asp:TreeNode Value = "Child3" Expanded = "True" Text = "3">
      <asp:TreeNode Value = "Grandchild1" Text = "A" />
    </asp:TreeNode>
  </Nodes>
</asp:TreeView>
```

② 在代码中绑定数据。

```
void initTreeView()
{
   SqlConnection con = new SqlConnection();
   con.ConnectionString = "Server= localhost;DataBase=NetBook;Integrated Security=True";
   con.Open();
   DataSet ds = new DataSet();
   SqlDataAdapter da = new SqlDataAdapter("select * from t_category", con);
   da.Fill(ds, "t_category");
   da = new SqlDataAdapter("select * from t_book", con);
   da.Fill(ds, "t_book");
   DataView dv_Categories = ds.Tables["t_category"].DefaultView;
   DataView dv_Books = ds.Tables["t_book"].DefaultView;
   //加入的是根节点
   TreeNode rootNode = new TreeNode();
   rootNode.ImageUrl = "~/images/1.gif";
   rootNode.Text = "图书浏览";
   TreeView1.Nodes.Add(rootNode);
   //加入子节点
   for (int i = 0; i < dv_Categories.Count; i++ )
   {
      TreeNode CategoryNode = new TreeNode();
      CategoryNode.ImageUrl = "~/images/Category.gif";
      CategoryNode.Text = dv_Categories[i]["CategoryName"].ToString();
      CategoryNode.Value = dv_Categories[i]["CategoryID"].ToString();
      rootNode.ChildNodes.Add(CategoryNode);
      dv_Books.RowFilter = "CategoryID=" + dv_Categories[i]["CategoryID"].ToString();
      for (int j = 0; j < dv_Books.Count; j++ )
      {
         TreeNode BookNode = new TreeNode();
         BookNode.ImageUrl = "~/images/Book.gif";
         BookNode.Text = dv_Books[j]["bookname"].ToString();
         BookNode.Value = dv_Books[j]["bookid"].ToString();
```

```
            CategoryNode.ChildNodes.Add(BookNode);
        }
    }
}
```

(8) 验证控件

验证控件包含验证逻辑,可以使用户对输入控件(例如 TextBox 控件)中输入的内容进行验证。验证控件可用于对必填字段进行检查,对照字符的特定值或模式进行测试,验证某个值是否在限定范围之内,等等。

ASP.NET 为用户提供了 6 个验证控件,它们分别是:RequiredFieldValidator、RangeValidator、RegularExpressionValidator、CompareValidator、CustomValidator 和 ValidationSummary,如图 3-5 所示。这些控件位于"工具箱"的"验证"面板中,它们均属于 Web 服务器控件,可以在 Web 页面中直接使用这些控件。

可以使用 ASP.NET 中的验证控件在 Web 窗体中验证用户的输入。验证控件可以提供常用的验证操作,例如比较两个值以及验证一个值是否位于范围内等。另外,也可以提供自己的验证操作并且显示自定义的错误信息。



图 3-5 验证控件

验证控件可以和任何服务器控件一起使用,而且每个验证控件都要引用一个服务器输入控件。处理用户的输入时,页面框架会把用户输入的内容传递给相应的验证控件,然后验证控件检查用户的输入并且根据检查结果设置是否正确设置相应的属性(指出验证是否通过)。在所有的控件验证都被调用之后,如果存在错误,则整个页面被设置为无效并显示相应的错误信息。

验证控件通常不显示在页面中,但是如果它们检测到错误,将产生指定的错误信息。这个错误信息可以用多种方式显示,通常把验证控件放在被验证控件的旁边,以就地显示错误信息。

下面列出了验证控件的一些常用属性。

① ControlToValidate 属性。该属性用来指定或获取将被验证的控件,即与该验证控件相关联的其他控件。

② Display 属性。该属性用来获取或设置控件显示错误信息的方式。可以指定如表 3-6 中所示的三种显示方式,其中默认为 Static 显示方式。

表 3-6 Display 显示方式

显示方式	说明
None	验证信息不显示,想要在 ValidationSummary 控件中集中显示验证信息时可以指定这种方式
Static	在页面布局中预留显示验证信息的空间,它属于页面的一部分
Dynamic	显示验证信息所需的空间被动态地添加到页面中,这种方式可以使多个验证空间处于相同的位置

③ EnableClientScript 属性。该属性表示是否是激活客户端的验证。如果该属性为 True,则执行客户端的验证,不论客户端验证是否处于激活状态,验证控件总是在服务器端

执行验证过程,但是客户端的验证不需要发送到服务器端进行处理,所以可以提高性能。

④ Enable 属性。该属性表示验证控件是否处于激活状态。可以使用这个属性来动态地激活或禁止某个验证过程。

⑤ ErrorMessage 属性。该属性用来获取或设置需要在 Web 页面上显示的验证控件的错误信息。

⑥ Is Valid 属性。该属性表示被验证的控件是否通过了验证。

ASP.NET 提供的 6 个验证控件介绍如下。

① RequiredFieldValidator 控件。RequiredFieldValidator 控件称为必须字段验证控件,其作用是保证与它相关联的控件中必须要输入内容才能通过验证。

在 RequiredFieldValidator 控件中包含一个 IntialValue 属性。这个属性用来指定相关联输入控件的初始值,其默认值为空字符串,即 string. Empty。指定初始值时,输入控件中的值必须与初始值不同才能通过验证。

通常使用这个控件验证是否在文本框中输入了信息,例如必须要输入用户名、密码的时候。

② CompareValidator 控件。CompareValidator 控件称为比较验证控件。在 ASP.NET 中,可以使用两种类型的比较验证控件:CompareValidator 控件和 RangeValidator 控件。这两种比较验证控件中都包含一个 Type 属性,Type 属性用来指定进行比较值的类型。

CompareValidator 验证控件可以用来将相关联的输入控件(通过 CompareValidator 的 ControlValidate 属性设置)的值与另一个控件的值(通过 CompareValidator 验证控件的 ControlToCompare 属性指定)进行比较。也可以将相关联的输入控件的值和一个指定的常量值(通过 CompareValidator 验证控件的 ValueToCompare 属性指定)进行比较。

在比较的时候,可以通过 CompareValidator 验证控件的 Operator 属性来指定比较操作的类型。表 3-7 给出了各种类型符号的说明,其中 Equal 为默认值。

表 3-7　Operator 属性设置及说明

Operator 属性	说　　明
Equal	当两个值相等时,验证通过
NotEqual	当两个值不相等时,验证通过
GreaterThan	当被验证控件的值大于指定的值或指定控件的值时,验证通过
GreaterThanEqual	当被验证控件的值大于等于指定的值或指定控件的值时,验证通过
LessThan	当被验证控件的值小于指定的值或指定的控件的值时,验证通过
LessThanEqual	当被验证控件的值小于等于指定的值或指定控件的值时,验证通过

③ RangeValidator 控件。RangeValidator 控件称为范围验证控件,它可以用来验证相关联的输入控件的值是否在指定范围内。可以通过 RangeValidator 验证控件的 MaximumValue 和 MinimumValue 属性来指定值的范围。

④ RegularExpressionValidator 控件。RegularExpressionValidator 控件称为正则表达式验证控件,它将检验被验证控件的值是否与指定的正则表达式相匹配。如果匹配,则通过验证;否则,验证不通过。使用这种验证控件,可以检查指定的字符串序列是否与指定的模式相匹配,例如 E-mail 地址、电话号码和邮政编码的格式是否正确等。

⑤ CustomValidator 控件。除了以上 4 种常用的验证控件外，ASP.NET 还支持自定义验证控件，自定义验证使用 CustomValidator 控件实现。使用这个控件，用户可以自定义输入控件的验证过程。

自定义验证控件的客户端的验证是通过一个 ClientValidationFunction 属性来实现的，这个属性用来指定自定义验证控件的客户端验证函数的名称。

自定义验证控件的服务器端的验证过程是通过响应它的 ServerValidate 事件来实现的，该事件参数中包含两个属性：

IsValid 属性：表示验证过程是否通过。如果通过，则在事件处理方法中把这个属性设为 True；否则设置为 False。

Value 属性：可以通过这个属性来获取被验证的值。

⑥ ValidationSummary 控件。ValidationSummary 控件用来总结页面中的所有验证错误，然后直接在页面中或者通过一个消息框来集中地显示它们。可以通过 ValidationSummary 控件的 DisplayMode 属性指定错误信息的显示方式，显示方式有表 3-8 所示的三种方式，其中 Bulletlist 为默认方式。

表 3-8 ValidationSummary 控件的显示方式

显示方式	说　　明
Bulletlist	以项目符号列表的方式显示页面中的所有验证控件的错误信息
List	列表的方式显示页面中的所有验证控件的错误信息
SingleParagraph	在一个段落中显示页面中的所有验证控件的错误信息

ValidationSummary 控件的 EnableClientScript 属性用来指定是否生成客户端的显示脚本；Forecolor 属性用来指定显示错误信息所使用的颜色；header text 属性用来指定标题头；而 ShowMessagebox 和 ShowSummary 属性则用来指定通过消息框显示错误信息还是直接在页面显示错误信息，可以把这两个属性都设置为 True，同时使用两种方式显示错误信息。

通过上面的介绍，了解了 ASP.NET 中验证控件的使用方法，下面就可以按照本项目任务要求，为用户注册页面添加验证。

3.4 项目实施

在接下来的内容里，将介绍子任务的实现过程。项目实施过程分解为如下的 3 个任务，每个任务由若干步完成。

3.4.1 任务 3-1 创建使用文本框、单选按钮、按钮能响应按钮事件的页面

1. 任务介绍

在这个任务中，计划设计一个用户能简单填写信息，并将信息提交到服务器的页面。

表 3-9 是目前在用的一个纸质样式的客户登记表。

表 3-9　客户登记表

NetBook 客户登记表			
姓名		性别	
出生日期		身份证	
手机		电话	
E-mail			
地址			
备注			

计划将该表设计成交互式的网页,用户可以通过页面填写自己的个人信息,并将信息提交到服务器上。

2. 任务分析

首先根据表格设计对应的数据库表,并且根据网络的应用特点,增加了用户名和密码两个字段。接着要做的就是设计一个类似注册的界面,并编写获取信息并存储到数据库的代码。

(1) 设计交互页面。在解决方案的 Web 文件夹中新建一个页面,打开该文件的设计界面,然后从“工具箱”的“标准”选项卡中选择相应的控件添加到设计界面中,如图 3-6 所示。

图 3-6　设计界面图

将密码和重复输入密码两个文本框的 TextMode 属性设置为 Password,将备注文本框的 TextMode 属性设置为 MultiLine。

考虑到用户名的字符串长度最好限制在 20 个字符之内以及密码的长度限制在 20 个字符之内,将用户名文本框的 MaxLength 属性设置为 20,密码和重复输入密码两个文本框的

MaxLength 属性设置为 20。

(2) 编写事件代码

界面基本设计好了，下面开始处理事件。选中 Button_Reg 按钮，双击鼠标，将会打开 UserRegister.aspx.cs 文件，并且会自动添加 Button_Reg _Click 事件函数。在 Button_Reg _Click 事件中输入以下代码，将页面注册信息写入数据库。

```
protected void Button_Reg _Click(object sender, EventArgs e)
{
    ///获取控件中的信息
    string ls_username = "",
            ls_name = "",
            ls_password1 = "",
            ls_password2 = "",
            ls_sex = "",
            ls_idcard = "",
            ls_email = "",
            ls_tel = "",
            ls_mobiletel = "",
            ls_memo = "";
    DateTime  ldt_birthday;
    ls_username = TextBox_UserName.Text;
    ls_password1 = TextBox_Password1.Text;
    ls_password2 = TextBox_Password2.Text;
    ls_name = TextBox_Name.Text;
    ls_idcard = TextBox_IDCard.Text;
    ls_email = TextBox_E-mail.Text;
    ls_tel = TextBox_Tel.Text;
    ls_mobiletel = TextBox_Mobiletel.Text;
    ls_memo = TextBox_Memo.Text;
        //出生日期必须进行类型转化
        ldt_birthday = Convert.ToDateTime(TextBox_Birthday.Text);

        //判断哪个单选按钮被选中
        if (RadioButton1.Checked)
            ls_sex = "男";
          else if (RadioButton2.Checked)
            ls_sex = "女";

        ///这段代码用于连接数据库,并将获取的用户信息保存到用户表中
        SqlConnection con = new SqlConnection();
        try
        {
            con.ConnectionString = "Server = localhost; DataBase = NetBook; Integrated
            Security=True";
            con.Open();

            SqlCommand command = new SqlCommand();
            command.Connection = con;
            command.CommandText = "insert into t_user (username, password, name, sex,
birthday, idcard, email, tel, mobiletel, memo) values(@username, @password, @name, @sex,
```

```
@birthday,@idcard,@email,@tel,@mobiletel,@memo)";

            command.Parameters.Add("@userName", SqlDbType.VarChar, 20).Value = ls_username;
            command.Parameters.Add("@password", SqlDbType.VarChar, 20).Value = ls_password1;
            command.Parameters.Add("@name", SqlDbType.VarChar, 20).Value = ls_name;
            command.Parameters.Add("@sex", SqlDbType.Char, 2).Value = ls_sex;
            command.Parameters.Add("@birthday", SqlDbType.DateTime).Value = ldt_birthday;
            command.Parameters.Add("@idcard", SqlDbType.Char, 18).Value = ls_idcard;
            command.Parameters.Add("@email", SqlDbType.VarChar, 100).Value = ls_email;
            command.Parameters.Add("@tel", SqlDbType.VarChar, 20).Value = ls_tel;
            command.Parameters.Add("@mobiletel", SqlDbType.VarChar, 20).Value = ls_mobiletel;
            command.Parameters.Add("@memo", SqlDbType.VarChar, 500).Value = ls_memo;
            command.ExecuteNonQuery();

            Label_Message.Text = "注册成功!";

        }
        catch (Exception ee)
        {
            Label_Message.Text = "注册失败!";
        }
        finally
        {
            con.Close();
        }
}
```

运行页面,在界面中输入信息,单击按钮,查看运行结果。

3. 任务完成总结

本任务中主要介绍使用文本框、单选按钮、按钮等简单控件来实现交互式页面。需要理解 ASP.NET 服务器控件是运行在服务器上的组件,它封装了相应的用户界面和相关功能,可以在 ASP.NET 页面文件和后台代码文件中使用。要重点掌握事件的处理方式。

4. 课堂训练与拓展

根据这个任务的设计过程,设计一个添加图书信息的交互式页面。

3.4.2　任务 3-2　创建带验证的页面

1. 任务介绍

任务 3-1 中设计的交互式页面基本能达到所要求的目标。但是在实际使用过程中输入身份证号码、邮政编码、日期或者电子邮件地址等信息时,常因为疏忽会将信息输错,例如把数字“0”输入成字母“O”,把数字“1”输成字母“l”,身份证号码漏掉一位等。因此希望能改进程序,让计算机能自动检查出这些错误,并提醒用户纠正输入。本任务中将介绍使用验证控件进行页面纠错检验。

2. 任务分析

根据前期设计的注册页面，开发人员希望在输入身份证号码、邮政编码、日期或者电子邮件地址等信息时，让计算机能自动检查出这些错误，并提醒用户纠正输入。这样可以减少错误的客户信息存储到后台数据库中，此外也可以间接帮助客户更好地完成信息录入，起到提示向导的作用。依据任务需求，将要验证的信息列成一张表，如表 3-10 所示。

表 3-10 控件验证分析

验证对象	验 证 要 求	解 决 方 案
用户名	不能为空	使用 RequiredFieldValidator 控件
密码	不能为空	使用 RequiredFieldValidator 控件
重复输入密码	必须和第一次输入的密码一致	使用 CompareValidator 控件，比较密码文本框和重复输入密码文本框的字符串是否一致
姓名	不能为空	使用 RequiredFieldValidator 控件
出生日期	必须是一个正确的日期格式	考虑到用户的出生日期应该和当前日期有一定的约束关系，例如如果现在是 2010 年，那用户输入的年龄应该在 1910～2010 年之间比较合理。因此使用 RangeValidator 控件，验证日期是否在合理范围内
身份证	必须是一个正确的身份证格式	使用 RegularExpressionValidator 控件，利用定义好的身份证号码格式
E-mail	必须是一个正确的 E-mail 格式	使用 RegularExpressionValidator 控件，利用定义好的 E-mail 格式

(1) 将验证控件加入到页面

依据表 3-11 的操作步骤将验证控件添加到页面。

表 3-11 验证操作步骤

验 证 对 象	操 作 步 骤
用户名	1. 加入 RequiredFieldValidator 控件 2. 选择 TextBox_UserName 作为 ControlToValidate 属性的值 3. 将 ErrorMessage 属性的值设置为“用户名不能为空”
密码	1. 加入 RequiredFieldValidator 控件 2. 选择 TextBox_Password1 作为 ControlToValidate 属性的值 3. 将 ErrorMessage 属性的值设置为“密码不能为空”
重复输入密码	1. 加入 CompareValidator 控件 2. 选择 TextBox_Password2 作为 ControlToValidate 属性的值 3. 选择 TextBox_Password1 作为 ControlToCompare 属性的值，即要求和 TextBox_Password1 的值进行比较，由于是比较两个文本框的字符串是否一致，因此 Operator 属性和 Type 属性保持默认设置 4. 将 ErrorMessage 属性的值设置为“两次输入的密码不一致”
姓名	1. 加入 RequiredFieldValidator 控件 2. 选择 TextBox_Name 作为 ControlToValidate 属性的值 3. 将 ErrorMessage 属性的值设置为“用户名不能为空”

续表

验 证 对 象	操 作 步 骤
出生日期	1. 加入 RequiredFieldValidator 控件 2. 选择 TextBox_Birthday 作为 ControlToValidate 属性的值 3. 考虑到比较的是日期，因此选择 Date 作为 Type 属性的值 4. 考虑到用户的出生日期应该和当前日期有一定的约束关系，不能把 MaximumValue 属性和 MinimumValue 属性设置为固定值，因此在 Page_Load 事件中加入： RangeValidator1. MaximumValue=DateTime. Now. ToShortDateString()； RangeValidator1. MinimumValue = DateTime. Now. AddYears(－100). ToShortDateString()； 5. 将 ErrorMessage 属性的值设置为"出生日期值不合理"
身份证	1. 加入 RegularExpressionValidator 控件 2. 选择 TextBox_IDCard 作为 ControlToValidate 属性的值 3. 单击 ValidationExpression 属性设置对话框，选择"中华人民共和国身份证号码(ID号)"作为正确表达式的值 4. 将 ErrorMessage 属性的值设置为"身份证格式不正确"
E-mail	1. 加入 RegularExpressionValidator 控件 2. 选择 TextBox_IDCard 作为 ControlToValidate 属性的值 3. 单击 ValidationExpression 属性的设置对话框，选择"Internet 电子邮箱地址"作为正确表达式的值 4. 将 ErrorMessage 属性的值设置为"E-mail 格式不正确"

加入验证控件后的页面设置效果如图 3-7 所示。

用户注册		
用户名：		用户名不能为空
密码：		密码不能为空
重复输入密码：		两次输入的密码不一致
姓名：		姓名不能为空
性别：	○男 ○女	
出生日期：		出生日期值不合理
身份证：		身份证格式不正确
手机：		
电话：		
E-mail：		E-mail格式不正确
备注：		
	注 册	
	[Label_Message]	

图 3-7 加入验证控件的页面设计界面

经过测试，添加了验证控件的页面基本达到了预期的要求，但是偶尔发现如果密码文本框里输入字符串，而在重复输入密码文本框中如果不输入内容，验证也能通过。这说明文本框里没有内容时，CompareValidator 验证控件是不起作用的。为了解决这个应用上的 Bug，需要为重复输入密码文本框再增加一个用于判断是否内容为空的 RequiredFieldValidator 控件，如图 3-8 所示。

结果问题得到了解决，这说明可以将多个验证控件同时作用在一个验证对象上。

重复输入密码： 重复输入密码不能为空 两次输入的密码不一致

图 3-8 重复输入密码文本框对应两个验证控件

（2）创建自定义验证

在设计注册用户登记表时，要求注册用户的用户名必须唯一，将用户名作为数据库中用户表的主键，主键不允许出现重复值。如果新用户在注册时使用的用户名与已注册过的用户的用户名相同，这将导致用户注册失败。由于验证用户名是否重复需要涉及访问数据库，并且需要设计一段验证的代码，之前用的验证控件无法胜任，因此可以使用自定义验证控件 CustomValidator 进行解决。

在用户名文本框后加入一个 CustomValidator 控件，如图 3-9 所示。



图 3-9 加入用户自定义控件

选择文本输入控件 TextBox_UserName 作为 ControlToValidate 属性的值。将 ErrorMessage 属性的值设置为“用户名已被注册”。

在 CustomValidator 控件的事件中选择 ServerValidate 事件，加入如下验证代码：

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    SqlConnection con = new SqlConnection();
    try
    {
        con.ConnectionString = "Server= localhost;DataBase=NetBook;Integrated Security=True";
        con.Open();
        SqlCommand command = new SqlCommand();
        command.Connection = con;
        command.CommandText = "select count(*) from t_user where username=@userName";
        command.Parameters.Add("@userName", SqlDbType.VarChar, 20).Value = args.Value;

        int n=(int)command.ExecuteScalar();
        //如果 n>0 表示数据库里已经有该用户名的记录,因此验证结果设置为假,否则验证结果设置为真
        if(n>0)
            args.IsValid=false;
        else
            args.IsValid=true;
    }
    catch (Exception ee)
    {
        args.IsValid = false;
    }
    finally
    {
        con.Close();
    }
}
```

在注册按钮的事件中加入一段页面内的验证是否都通过的代码：

```
protected void Button_Reg_Click(object sender, EventArgs e)
{
   if (Page.IsValid == false)
   {
       return;
   }
   //下面代码省略
}
```

至此，验证用户名是否重复的问题得以解决。

（3）验证控件测试

从三个方面来测试验证控件：不输入任何信息、输入错误信息、输入重复用户名。

① 什么都不输入，验证结果如图 3-10 所示。

② 输入错误密码或输入错误的出生日期、身份证、E-mail 等信息，验证结果如图 3-11 所示。

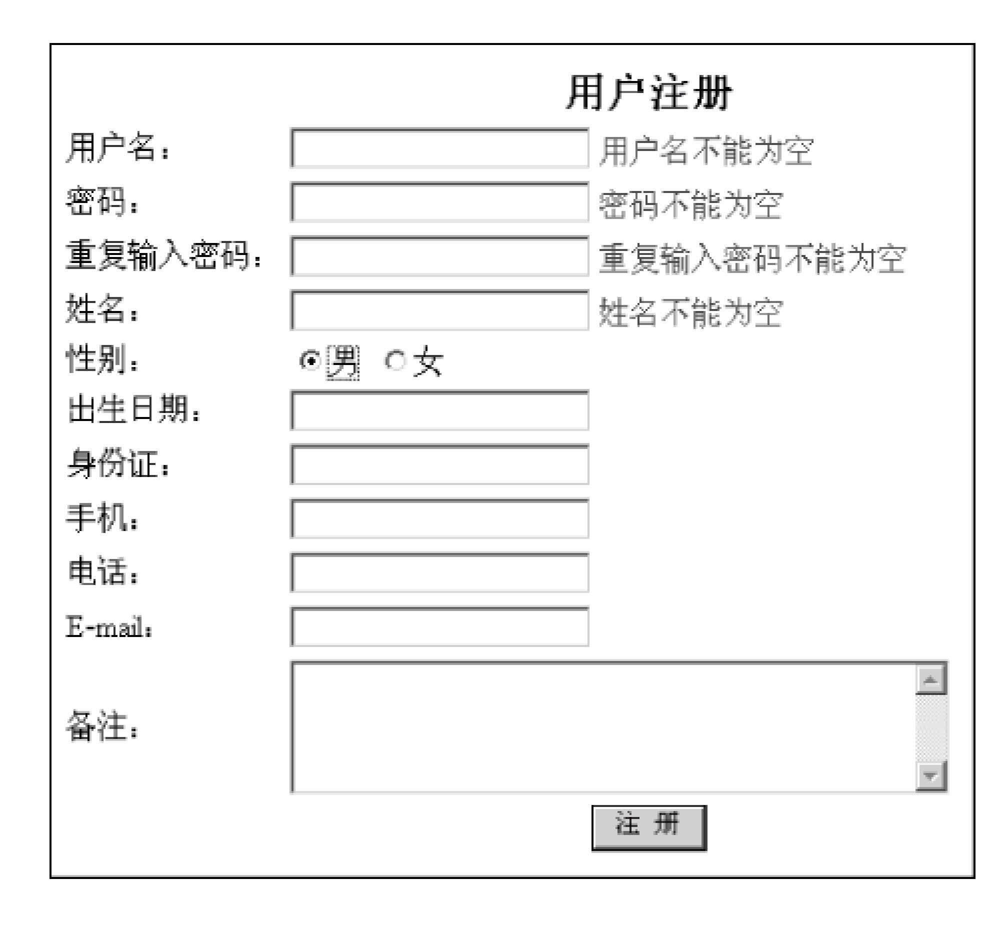


图 3-10　不输入任何内容的验证测试

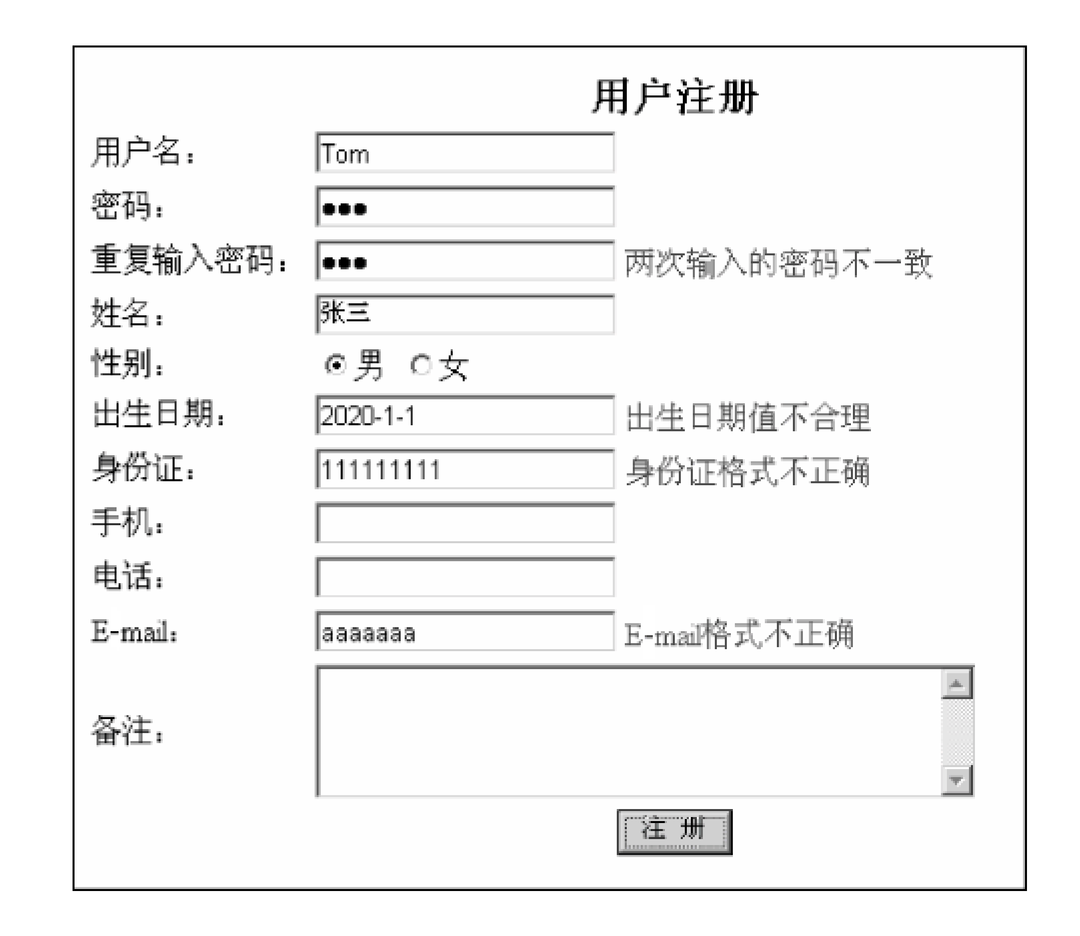


图 3-11　输入错误信息的验证

③ 输入数据库中已有的用户名，验证结果如图 3-12 所示。

用户注册
用户名：Tom 用户名已被注册
密码：
重复输入密码：
姓名：Tom
性别：男 女
出生日期：2000-1-1
身份证：
手机：
电话：
E-mail：
备注：
注 册

图 3-12　重复用户名验证

3. 任务完成总结

在网站中，对输入数据进行校验是最常用的技术之一。在 ASP.NET 2.0 中，校验工作在服务器端进行，在可能的情况下，将自动调用客户端验证作为补充，以减少错误信息在网络上往返的次数，提高使用效率。

本任务介绍了常用的 5 种验证控件，包括了 RequiredFieldValidator、CompareValidator、RangeValidator、RegularExpressionValidator、CustomValidator。尽管这些控件的作用不一样，但是使用方法却有着很多共同点，都需要将属性指向被验证的控件，指定错误发生时提示的语句，其他属性的设置则根据控件的作用不同而有所不同。这些控件除了 RequiredFieldValidator 控件外，其他控件都允许空的输入，因此需要将此控件与其他控件一起指向输入控件时，才能避免输入错误。

4. 课堂训练与拓展

使用本任务中学习的验证控件，完善前面设计好的图书信息交互式页面，使修改的页面在保存图书信息时，能验证输入的不规范信息。

3.4.3 任务 3-3 创建使用下拉菜单、上传控件的页面

1. 任务介绍

在录入用户的注册信息时，希望能够在客户的信息中增加一个填写所在省份的信息栏，以便了解客户的分布情况。此外，为保存完整的用户信息，希望能够实现客户发送照片到服务器。这两个任务可以使用下列控件和文件上传控件实现。

本任务中将实现在客户的信息中增加一个填写所在省份的信息栏，以及实现客户发送照片到服务器的功能。

2. 任务分析

要增加一个填写所在省份的信息栏，如果只是增加一个文本框，填写内容应该很容易实现，但是考虑到这个信息将用于统计各省份的客户人数，如果随意让用户填写，将很难统计准确。例如，用户来自广东，可能填写广东、广东省、粤等各种形式。因此，为规避上述可能出现的情况，使用下拉菜单将所有的省份以选项的形式列出，由用户选择所在省份，这样消除了同一省份写法各异的问题。

从客户端通过网络发送照片到服务器，可以通过上传控件实现。

(1) 使用下拉菜单收集用户省份信息

首先，在用户表中增加一个省份字段(state)。

然后，将 DropDownList 下拉控件放入页面中，如图 3-13 所示。打开其 Items 属性，添加 ListItem 项，设置 ListItem 项的 Text 和 Value 属性为加入的省份名称，如图 3-14 所示。

最后修改代码，我们必须获取下拉菜单的选中项，并和其他信息一起保存到数据库，添加的获取下拉菜单的选中项部分如下：



图 3-13　加入 DropDownList 控件的页面



图 3-14　添加省份选项

```
string  ls_state;
ls_state = DropDownList1.SelectedItem.Value;
```

(2) 使用上传控件

首先,在用户表中增加一个照片字段(photo)。

然后,将上传控件、按钮以及两个标签放入页面中,如图 3-15 所示。按钮的作用是单击以后可以触发执行上传图片的事件。第一个标签“Label_Upload”用于显示上传是否成功的信息;第二个标签“Label_FileName”用于记录上传后的文件名,以便在单击“注册”按钮保存信息时,可以通过该标签获取文件名称。由于“Label_FileName”标签只是起到操作过程中记录文件名的作用,因此将它的 Visible 属性设置为 False。

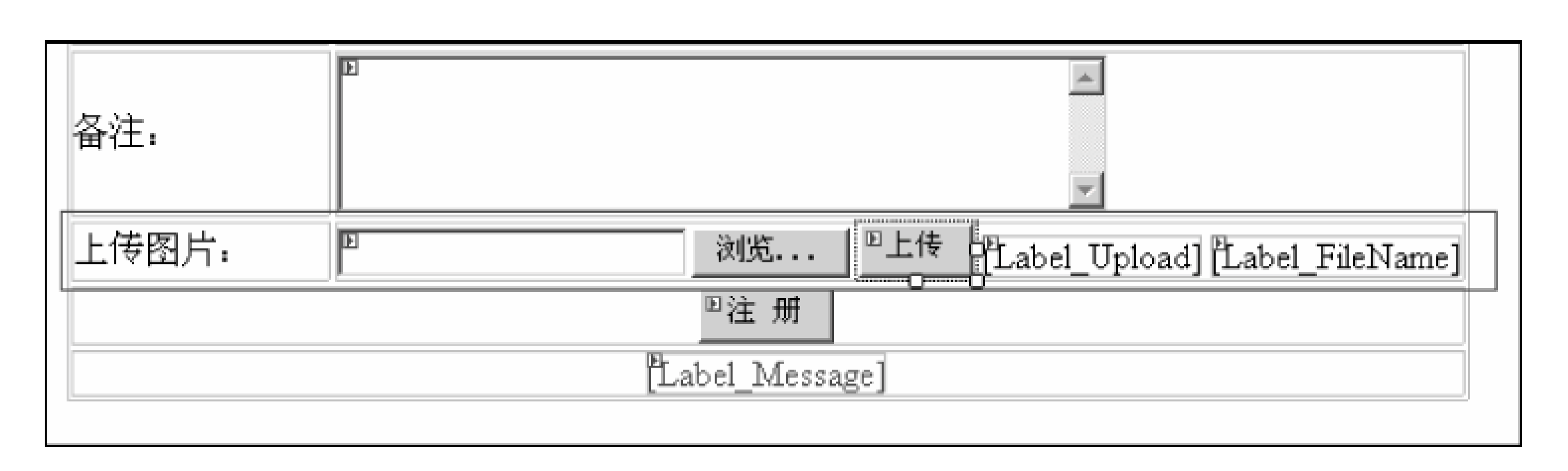


图 3-15　加入 FileUpload 控件的页面

接下来编写单击“上传”按钮的事件代码,如下:

```
protected void Button_Upload_Click(object sender, EventArgs e)
{
   try
   {
```

```
        //上传文件的文件名(含完整路径)
        string fileName = FileUpload1.PostedFile.FileName;

        //上传文件的大小(byte)
        int fileLength = FileUpload1.PostedFile.ContentLength;

        //取文件的扩展名
        string fileType = fileName.Substring(fileName.LastIndexOf(@"."));

        //考虑到要用户上传的文件名有可能会同名,导致前面的文件被覆盖,我们可以将文件名重
新设置成利用时间函数创建的唯一名称
        string Randname = DateTime.Now.ToString("yyyyMMddhhmmss") + DateTime.Now.Ticks.ToString();

        //完整的新文件名
        string newfileName = Randname + fileType;

        //使用 SaveAs 方法,将文件保存在项目路径\upload 目录下
         FileUpload1.PostedFile.SaveAs(HttpRuntime.AppDomainAppPath + @"\upload\" +
newfileName);

        Label_FileName.Text = newfileName;
        Label_Upload.Text = "成功上传文件:" + fileName + ",文件大小:" + fileLength +
"字节," + "文件类型:" + FileUpload1.PostedFile.ContentType;

    }
    catch (Exception ee)
    {
        Label_Upload.Text = "上传文件失败";
    }
}
```

运行页面,先上传图片再输入信息,单击按钮,查看运行结果。

3. 任务完成总结

本任务主要介绍了使用下拉控件和上传控件实现一个较复杂的页面。读者在学习了 DropDownList 后可以试着使用 ListBox、RadioButtonList、CheckBoxList、BulletedList 等控件,因为这些控件和 DropDownList 有着类似的特性和功能。

4. 课堂训练与拓展

在图书信息的交互式页面中加入能选择图书种类的下拉菜单,以及增加一个可以上传图书封面图片的功能。

3.5 项 目 总 结

本项目结合实际任务介绍了常用的 Web 服务器控件使用方法。Web 服务器控件中主要介绍了 Label 控件、TextBox 控件、RadioButton 控件、Button 控件、DropDownList 控件、

FileUpload 控件以及常用的 6 种验证控件，包括 RequiredFieldValidator 控件、CompareValidator 控件、RangeValidator 控件、RegularExpressionValidator 控件、CustomValidator 控件、ValidationSummary 控件。由于任务要求的限制，其他 Web 服务器控件本案例中不再介绍，读者可以查看一些其他介绍控件使用方式的配套书籍学习。

在学习控件使用的方法上，可以采取先了解其有何用途，然后学习该控件的常用属性，学习时可以采取属性窗口和代码编写两种方式对照进行，最后要学习控件的一些重要事件。

本项目要求读者掌握常用的服务器控件的使用方法。

3.6 项目实训

1. 任务描述

把图书信息通过数据库来管理，能够在网上添加图书信息。

2. 任务要求

① 根据图 3-16 所示创建一张图书表。

列名	数据类型	长度	允许空
bookid	int	4	
bookname	varchar	100	✓
ISBN	varchar	50	✓
author	varchar	50	✓
publisher	varchar	50	✓
pubdate	datetime	8	✓
price	decimal	9	✓
picture	varchar	500	✓
memo	varchar	1000	✓

图 3-16 图书表

② 设计一个添加图书的页面，其中出版社(publisher)要求以下拉菜单的方式选择。

③ 对书名(bookname)、出版号(ISBN)、出版时间(pubdate)、价格(price)进行验证，其中书名(bookname)、出版号(ISBN)不能为空，出版时间(pubdate)必须是正确日期，价格(price)必须是大于 0 的数字。

④ 能够上传图书封面图片，并将图片文件名保存在 picture 字段中。

项目4

实现网上书店图书查询功能

4.1 项 目 介 绍

在本项目中,我们将实现浏览图书的信息和查阅特定图书的功能,这些图书信息都保存在 SQL Server 数据库中,因此本项目的核心内容有以下两方面:

① 联接数据库,查询数据库中的有关内容。

② 使用数据显示控件呈现数据。

4.2 项 目 分 析

大多数有组织的数据都以关系型数据库的某种格式保存,例如 Microsoft SQL Server 和 Oracle,这些系统具有可伸缩性、健壮性以及可管理等特性,可以支持最大和最繁忙的 Web 站点。

网上书店的业务信息都存放在 SQL Server 数据库 NetBook 中,它以一个图书销售公司为模型,内容完整,本书以后对数据库的访问部分,就以它为例子。

表 4-1 列出了 NetBook 数据库的所有用户表,主要有表示书籍的 t_book、表示用户的 t_user、表示目录的 t_category 等。表 4-2～表 4-4 分别列出了表 t_user、t_book、和 t_category 的定义。

表 4-1 数据库 NetBook 中的用户表

库 名	表 名	功 能 说 明
NetBook	t_user	存储用户信息
	t_address	用于记录客户的地址信息
	T_appraise	记录客户的评价信息
	t_book	记录书籍的详细信息
	t_category	存储图书的种类
	t_cart	存储购物车信息
	t_order	存储用户订单信息
	t_orderdetail	存储订单详细信息
	t_research	存储调查信息
	t_SecondCategory	用于存储图书种类信息
	t_news	存储图书新闻信息

续表

库　　名	表　　名	功 能 说 明
NetBook	t_ researchItem	用于记录用户调查的详细信息
	t_consultation	用于记录用户询问相关情况的信息
	P_Group	用户组
	P_GroupMenu	组对应的菜单
	P_LoginUser	登录用户
	P_Menu	菜单列表
	P_UserGroup	用户信息

表 4-2　t_user 表的定义

表名：t_user(用户基本信息)					
列　　名	描　　述	数据类型(精度范围)	空/非空	唯一	描述
name	用户真实姓名	Varchar(20)	非空		
password	用户密码	Varchar(20)	非空		
sex	性别	Char(2)	非空		
birthdate	出生日期	Datetime(8)			
email	电子邮件	Varchar(100)			
tel	联系方式	Varchar(20)	非空		
address	收货地址	Varchar(50)	非空		
mobiletel	手机号码	Varchar(20)			
memo	备注	Varchar(500)			
zipcode	邮政编码	Char(6)			
receiver	收货人	Varchar(20)			
其他说明	Primary Key：name				

表 4-3　t_book 表的定义

表名：t_book(图书基本信息)					
列　　名	描　述	数据类型(精度范围)	空/非空	唯一	约束条件
bookid	图书 ID	Int(4)	非空	唯一	
bookname	图书书名	Varchar(20)	非空		
ISBN	图书出版号	Varchar(50)			
author	作者 Id	Int(4)	非空	唯一	
publisher	出版社	Varchar(50)			
pubdate	出版时间	Datetime(8)			
price	价格	Money(8)	非空		
NetBookprice	网价	Money(8)			
Picture	图片	Varchar(500)			
memo	备注	Varchar(1000)			
recommendFlag	是否推荐	Bit(1)			
recommendDate	推荐日期	Datetime(8)			
recommendMemo	推荐备注	Varchar(1000)			
amount	图书库存	Int(4)			

续表

列　　名	描　　述	数据类型(精度范围)	空/非空	唯一	约束条件
NetBookDate	上架时间	Datetime(8)			
secondCategoryId	种类编号	Int(4)			
CategoryId	种类 Id	Int(4)			
discount	折扣	Money(8)			
其他说明	Primary Key：bookid Foreign Key：AuthorId,CategoryId				

表 4-4　t_category 表的定义

表名：Category(图书种类)					
列　　名	描　　述	数据类型(精度范围)	空/非空	唯一	描述
CategoryId	图书种类 ID	Int(4)	非空	唯一	
CategoryName	种类名称	Varchar(50)	非空		
其他说明	Primary Key：CategoryId				

本项目的核心任务是联接数据库和显示数据。对于联接到 SQL Server 数据库,我们可以使用控件 SqlDataSource 来实现,而控件 GridView 是显示数据的典型工具,提供了非常丰富且强大的功能。

4.3　相关知识

1. 数据库连接控件 SqlDataSource

通过使用数据源控件和数据绑定技术,可以非常方便地访问数据库,在页面上显示其中的数据,或者编辑、增删数据记录。

数据源控件即是封装了数据库操作功能的可视组件,包括:

- SqlDataSource:这个控件可以让我们连接到任何具有 ADO.NET 数据提供程序的数据源,包括 SQL Server,Oracle 和其他 ODBC 数据源。即使不写任何数据访问代码,也能方便地使用这个数据源控件,它也是本书介绍的重点。
- ObjectDataSource:这个控件可以让我们连接到一个自定义数据访问类,通常在大型网站中使用。
- XmlDataSource:这个控件可以让我们连接到一个 XML 文件。
- SiteMapDataSource:这个控件可以让我们连接到 Web. Sitemap 文件,对整个网站提供导航结构。

在 Visual Studio 工具箱的 Data 面板中可以找到所有与数据相关的控件。

要了解在 ASP.NET 2.0 中,如何通过 Web 页面访问数据库,我们首先需要了解 ASP.NET 2.0 是如何对数据进行处理的,图 4-1 以 SqlDataSource 控件为例,给出了 ASP.NET 2.0 中的数据处理架构。

由图 4-1 可以看出,ASP.NET 2.0 使用 SqlDataSource 控件作为 Web 页面与数据库联

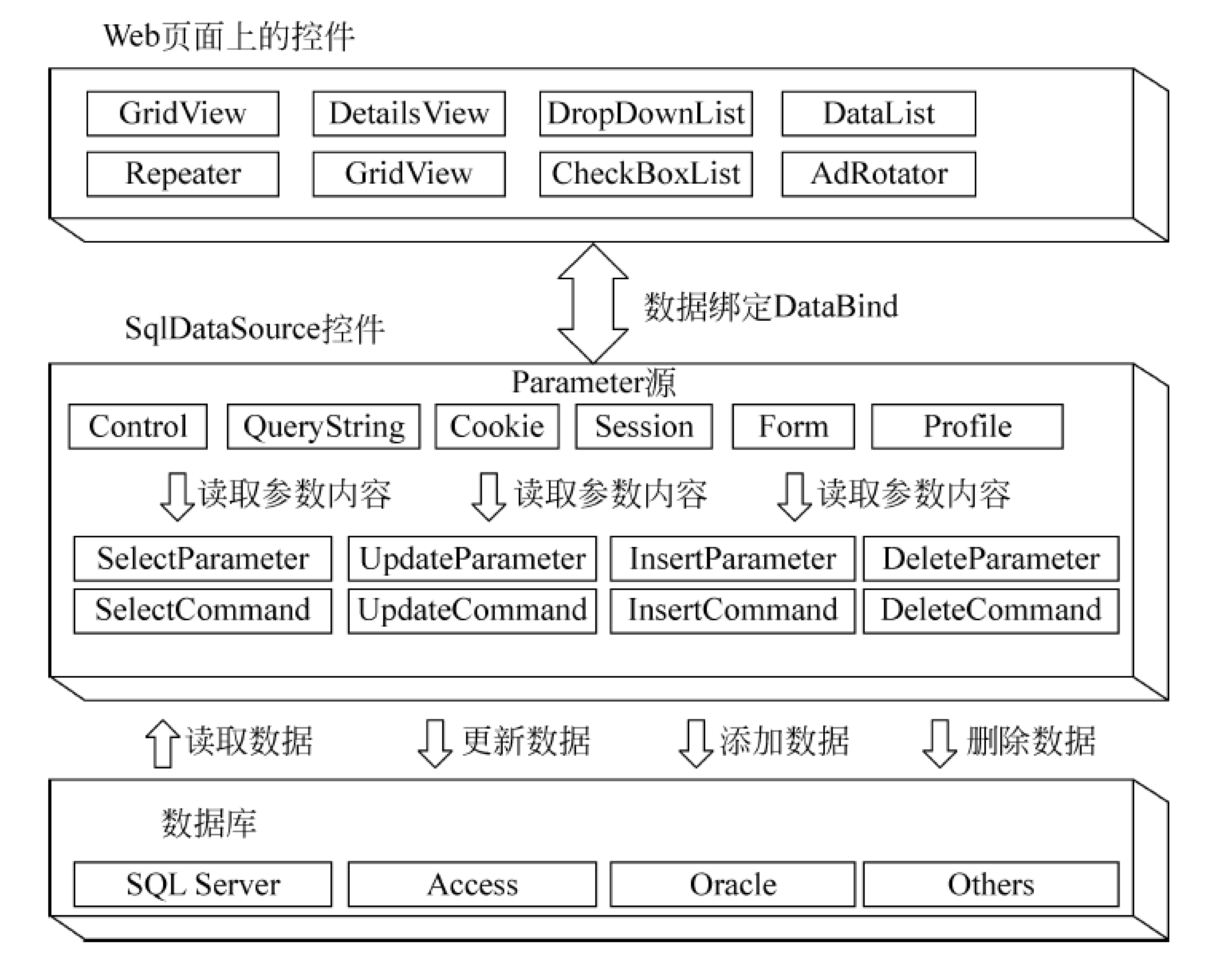


图 4-1　ASP.NET 2.0 中数据处理架构

系的桥梁，使得我们在程序中访问数据库时，不必考虑“如何访问”的问题，而可以专注于解决“访问什么”的问题。SqlDataSource 控件提供了 Web 页面与数据库联系的双向功能，Web 控件可以通过数据绑定命令与 SqlDataSource 控件关联，而 SqlDataSource 控件可以通过 SelectCommand、UpdateCommand、InsertCommand、DeleteCommand 对各种数据库进行相应操作。

在.aspx 文件中，数据源控件就和普通控件一样出现，例如：

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server" ... />
```

在 SqlDataSource 控件中，特别要关注的就是连接字符串和 SQL 命令，它们都可以通过 SqlDataSource 的配置向导来设置，也可以在.aspx 文件中直接编写。

SqlDataSource 控件的连接字符串包含了连接的数据源名称、安全认证模式和用户名密码等重要信息，它可以在 SqlDataSource 控件中显式定义，如下所示：

```
<asp:SqlDataSource ID = "SqlDataSource1"
    runat = "server"
    ConnectionString = "Data Source = localhost;Initial Catalog = MyPubs;Integrated Security = True"
    ProviderName = "System.Data.SqlClient" >
</asp:SqlDataSource>
```

为了使程序易于移植和修改，通常情况下是把连接字符串存放在配置文件 web.config 中，SqlDataSource 控件再从中读取。web.config 文件将包含类似如下示例的片段：

```
<configuration>
    <connectionStrings>
        <add name = "HuaLongConnectionString"
    connectionString = "Data Source = localhost;Initial Catalog = NetBook;Integrated Security = True"
```

```
        providerName = "System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

上面的示例定义了一个连接字符串，它的名字是 HuaLongConnectionString，值是"Data Source=localhost;Initial Catalog=NetBook;Integrated Security=True"。有了以上的配置信息，SqlDataSource 控件就可以直接使用它了，具体如下所示：

```
<asp:SqlDataSource ID = "SqlDataSource1"
      runat = "server" ConnectionString = "<% $ ConnectionStrings:HuaLongConnectionString %>"
...>
</asp:SqlDataSource>
```

一旦我们配置好 SqlDataSource 的数据库连接信息后，下一步就是添加查询逻辑，告诉 SqlDataSource 将要对数据库作何种操作。SqlDataSource 使用的查询逻辑，既可以是 SQL 语句，也可以是数据库中的存储过程。在这些 SQL 语句或存储过程中，还可以使用参数，这些参数的具体数值，可以由控件、查询字符串、Session 状态等多种形式来提供。完整的使用 SQL 语句的 SqlDataSource 控件的语法形式如下所示：

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
      ConnectionString = "<% $ ConnectionStrings:连接字符串 %>"
      DeleteCommand = "DELETE FROM ... WHERE ..."
      InsertCommand = "INSERT INTO ...  VALUES ..."
      UpdateCommand = "UPDATE ... SET ... WHERE..."
      SelectCommand = "SELECT ... FROM ... WHERE..."
      <DeleteParameters>
          ⋮
      </DeleteParameters>

      <InsertParameters>
          ⋮
      </InsertParameters>
      <UpdateParameters>
          ⋮
      </UpdateParameters>
      <SelectParameters>
          ⋮
      </SelectParameters>
</asp:SqlDataSource>
```

这四种查询命令并不是必需的，在通常情况下，我们可能只需要使用其中的部分功能，而省略掉其他部分。

2. 数据显示控件 DataList

DataList 控件以某种格式显示数据，这种格式可以使用模板和样式进行定义。DataList 控件可用于任何重复结构中的数据。DataList 控件可以以不同的布局显示行，如按列或按行对数据进行排序。DataList 控件还可以被配置为允许用户编辑或删除信息，也可以自定义该控件以支持其他功能，如选择行。

表 4-5 列出了 DataList 控件的模板。

表 4-5　DataList 控件的模板

模　板　名	说　　明
ItemTemplate	必须定义，每一项的内容和布局的默认定义
AlternatingItemTemplate	为每一个间隔项提供内容和布局(默认为 ItemTemplate)
EditItemTemplate	为当前正在编辑的项提供内容和布局(默认为 ItemTemplate)
SelectedItemTemplate	为当前选中的行提供内容和布局(默认为 ItemTemplate)
FooterTemplate	为页脚提供内容和布局
HeaderTemplate	为标题提供内容和布局
SeparatorTemplate	为项与项之间的分隔符提供内容和布局

若要在模板中指定项的外观，可以设置该模板的样式。例如，我们可以指定以下样式：

- 在白色背景上用黑色文本呈现各项。
- 在浅灰色背景上用黑色文本呈现交替项。
- 在黄色背景上用黑色加粗文本呈现选定项。
- 在浅蓝色背景上用黑色文本呈现正在编辑的项。

每个模板支持其自己的样式对象，可以在设计和运行时设置该样式对象的属性。

DataList 控件使用 HTML 表对应用模板的项进行布局，从而可以控制各个表单元格的顺序、方向和列数，这些单元格用于呈现 DataList 项。表 4-6 描述了 DataList 控件支持的布局选项。

表 4-6　DataList 控件支持的布局选项

布 局 选 项	说　　明
流布局	在流布局中，列表项在行内呈现，如同文字处理文档一样
表布局	在表布局中，列表项在 HTML 表中呈现。由于在表布局中可以设置表单元格属性(如网格线)，这就为我们提供了更多可用于指定列表项外观的选项
垂直布局和水平布局	默认情况下，DataList 控件中的项在单个垂直列中显示。但是，可以指定该控件包含多个列。如果这样，可进一步指定这些项是垂直排序(类似于报刊栏)还是水平排列(类似于日历中的日)
列数	不管 DataList 控件中的项是垂直排序还是水平排序，我们都可指定列表将有多少列。这使我们能够控制网页呈现的宽度，通常可避免水平滚动

DataList 控件支持多种事件。其中，ItemCreated 事件可以在运行时自定义项的创建过程。ItemDataBound 事件提供了自定义 DataList 控件的能力，但需要在数据可用于检查之后。例如，如果正使用 DataList 控件显示任务列表，则可以红色文本显示过期项，以黑色文本显示已完成项，以绿色文本显示其他任务。这两个事件都可用于重写来自模板定义的格式设置。

其余事件为了响应列表项中的按钮单击而引发，这些事件旨在帮助我们响应 DataList 控件的最常用功能。支持该类型的 4 个事件有 EditCommand、DeleteCommand、UpdateCommand 和 CancelCommand。若要引发这些事件，可将 Button、LinkButton 或 ImageButton 控件添加到 DataList 控件中的模板中，并将这些按钮的 CommandName 属性

设置为某个关键字，如 edit、delete、update 或 cancel。当用户单击项中的某个按钮时，就会向该按钮的容器(DataList 控件)发送事件。按钮具体引发哪个事件将取决于所单击按钮的 CommandName 属性的值。例如，如果某个按钮的 CommandName 属性设置为 edit，则单击该按钮时将引发 EditCommand 事件。如果 CommandName 属性设置为 delete，则单击该按钮将引发 DeleteCommand 事件，以此类推。

DataList 控件还支持 ItemCommand 事件，当用户单击某个没有预定义命令(如 edit 或 delete)的按钮时将引发该事件。我们可以按照如下方法将此事件用于自定义功能：将某个按钮的 CommandName 属性设置为一个自己所需的值，然后在 ItemCommand 事件处理程序中测试这个值。

3. 数据显示控件 GridView

GridView 控件非常适合用来显示处理表格数据，它与 SqlDataSource 控件结合，可以完成大部分数据处理的工作，包含增加、删除、修改、选择、排序等功能。在很多情况下，仅通过添加各种控件并设置好它们的属性后，就可以实现对数据库的一般访问。

GridView 控件显示成一个表格形式，外观和样式可以配置，图 4-2 展示了一个典型的例子。

title	type	price	
But Is It User Friendly?	popular_comp	￥22.95	选择
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	￥21.59	选择
Cooking with Computers: Surreptitious Balance Sheets	business	￥11.95	选择
Emotional Security: A New Algorithm	psychology	￥7.99	选择
Fifty Years in Buckingham Palace Kitchens	trad_cook	￥11.95	选择
Is Anger the Enemy?	psychology	￥10.95	选择
Life Without Fear	psychology	￥7.00	选择
Net Etiquette	popular_comp		选择
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	￥20.95	选择
Prolonged Data Deprivation: Four Case Studies	psychology	￥19.99	选择
1 2			

图 4-2 典型的 GridView 控件

表中的每一列为一个字段，最常见的是由数据库访问而得到的绑定字段，除此之外还有命令字段、按钮字段等，表 4-7 列出了 GridView 控件中可用的所有字段。

表 4-7 GridView 的字段

GridView 字段	说 明
BoundField(数据绑定字段)	将数据以文字方式显示，默认字段
ButtonField(按钮字段)	可显示为 PushButton 或 LinkButton，单击产生 RowCommand 事件
CheckBoxField(CheckBox 字段)	将数据以 CheckBox 控件方式显示，数据类型为 Boolean 时适用
CommandField(命令字段)	显示编辑、删除、修改、选择时的按钮
HyperLinkField(超级链接字段)	将数据以 HyperLink 超链接方式显示
ImageField(图片字段)	将数据以图片方式显示
TemplateField(模板字段)	模板字段，可将字段替换为任何控件，并实现数据绑定

GridView 控件根据字段的不同，其属性也不一样，但是有些属性是相同的，如表 4-8 所示。

每个字段可设置如表 4-9 所示的样式。

表 4-8　字段的共同属性

共同属性	说　明
FooterText	表尾文字
HeaderImageUrl	表头文字 Url
HeaderText	表头文字
SortExpression	设置排序字段

表 4-9　字段的共同样式

GridView 样式	说　明
FooterStyles	表尾样式
ItemStyles	项目样式
HeaderStyles	表头样式
ControlStyles	控件样式

(1) 使用智能标记

智能标记是 Visual Studio 2005 的新增功能，一些比较复杂的控件都提供智能标记，通过它可以快速设置控件。打开智能标记有下列两种方法。

方法一：单击图标，打开智能标记，如图 4-3 所示。

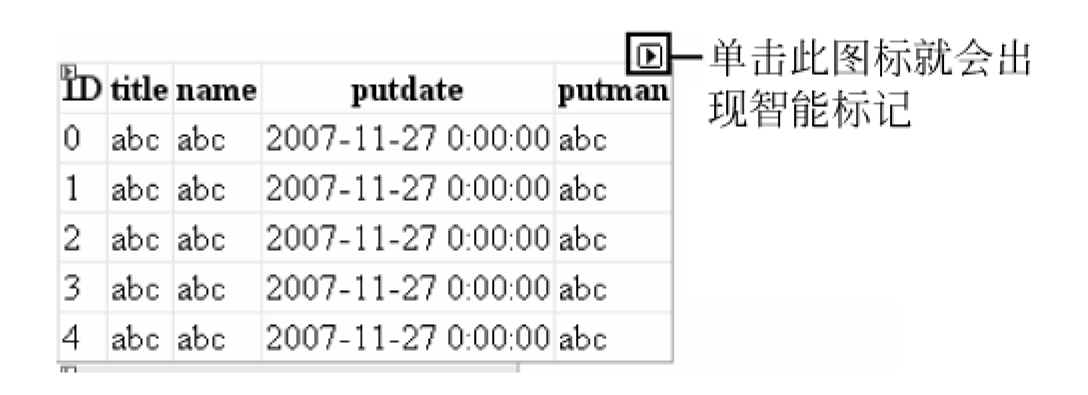


图 4-3　打开智能标记方法 1

方法二：右击，利用快捷菜单打开智能标记，如图 4-4 所示。

如果要关闭智能标记，只需要单击设计界面的其他地方即可。智能标记的功能如图 4-5 所示。

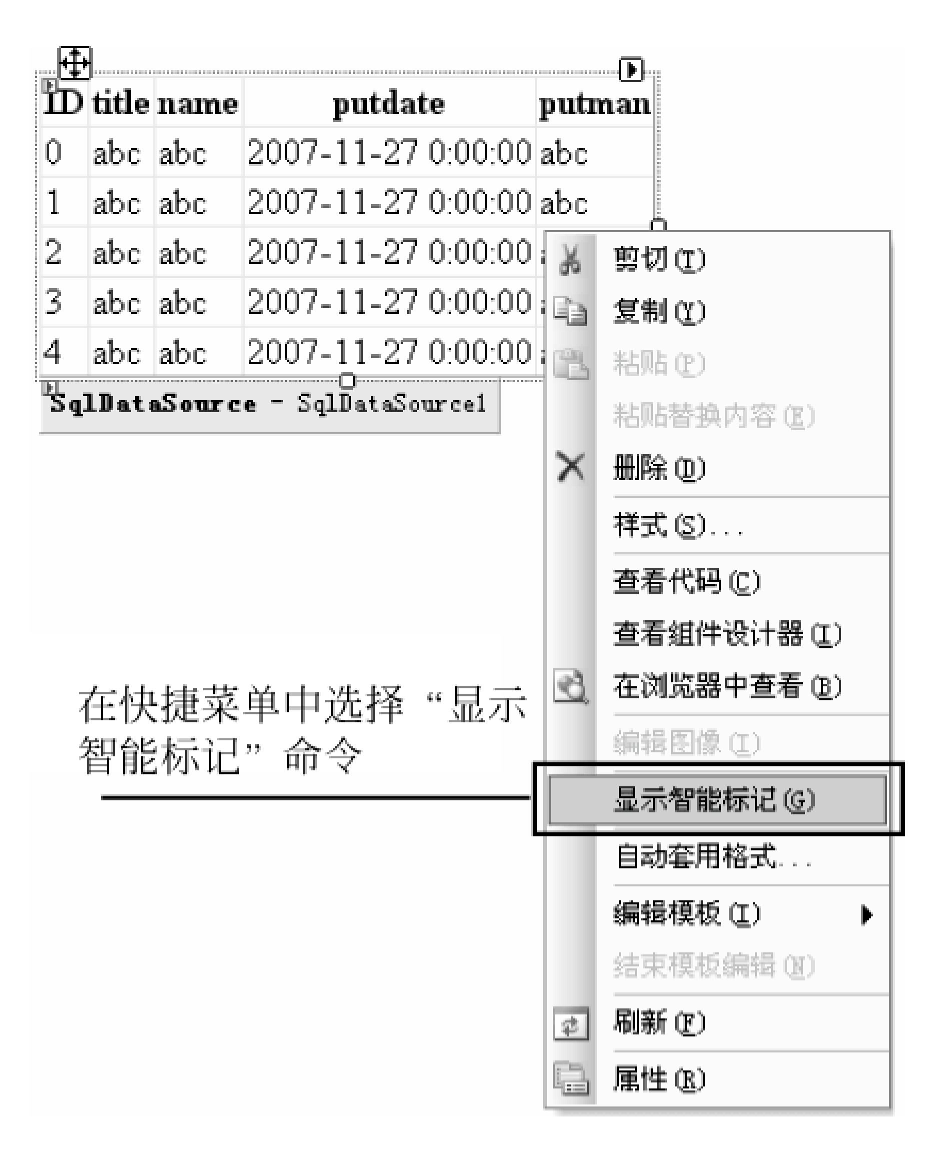


图 4-4　打开智能标记方法 2

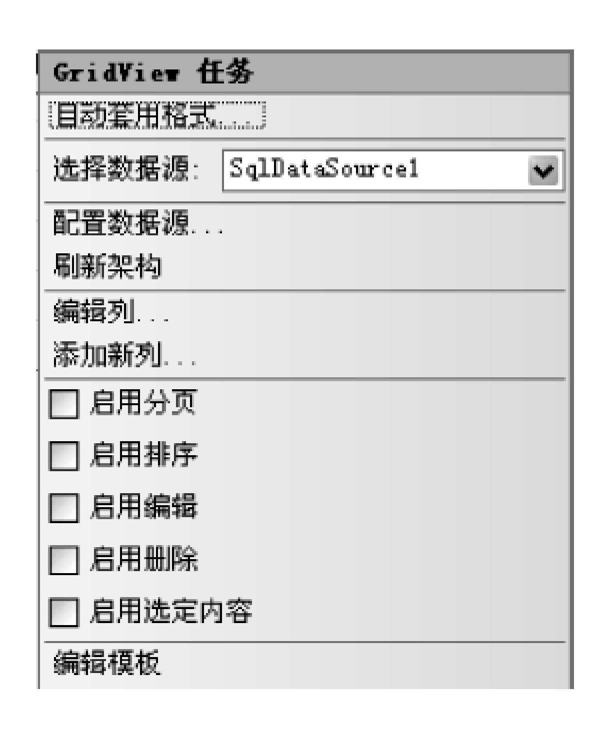


图 4-5　GridView 智能菜单

(2) 添加分页功能

当用 GridView 控件显示数据时，如果数据行数很多，就会使查阅变得很不方便，此时就可以使用分页功能将所有数据分页显示，每次只显示一页。对 GridView 添加分页功能非常简单，只需在智能标记中，选中“启用分页”复选框即可。在分页(PagerSettings)属性组中有很多属性可设置分页按钮的功能和外观，表 4-10 列出了与分页相关的主要属性。

表 4-10　分页的主要属性

属性名称	功　　能	属性名称	功　　能
AllowPaging	是否允许分页	NextPageText	下一页文字
PageIndex	分页索引	PageButtonCount	显示分页按钮的个数
FirstPageImageUrl	切换第一页图片	Position	分页位置
FirstPageText	切换第一页文字	PreviousPageImageUrl	上一页图片
LastPageImageUrl	切换最后一页图片	PreviousPageText	上一页文字
LastPageText	切换最后一页文字	Visible	是否显示分页
Mode	设置分页模式	PageSize	每一页包含的记录数量
NextPageImageUrl	下一页图片		

GridView 控件还可以用不同的分页模式来显示数据，这个模式由 Mode 属性设置，具体如表 4-11 所示。

表 4-11　分页模式

Mode 设置	说　　明	图　　例
NextPrevious	只有上一页、下一页	<>
Numeric	只显示数字	123456
NextPreviousFirstLast	显示第一页、上一页、下一页、最后一页	<<<>>>
NumericFirstLast	显示上一页、下一页、数字	<<…456…>>

分页按钮的位置由 Position 属性设置，具体如表 4-12 所示。

表 4-12　分页的 Position 属性

Position 属性设置	说　　明
Bottom	显示在下方
Top	显示在上方
TopAndBottom	显示在上方及下方

(3) 添加排序功能

GridView 控件的排序功能使用户只需单击 GridView 控件的字段标题，就可以对此字段进行排序。选中智能标记的“启用排序”复选框，即可添加排序功能。相应的控件属性是 AllowSorting 和 SortExpression。图 4-2 就是以字段 title 进行排序的，如果单击“type”或“price”，也会相应修改排序字段。

(4) 添加选择功能

在 GridView 控件中添加选择功能后，每一行都会出现选择按钮，用户单击行的选择按钮后会触发事件，可以添加事件代码响应用户的选择。要对 GridView 控件添加选择功能，只需在智能标记之内选中“启用选定内容”复选框即可。添加了选择功能后，系统会自动加入一个 CommandField，如下列代码所示：

```
<asp:CommandField ShowSelectButton = "True" />
```

（5）添加编辑功能

GridView 控件的编辑功能可以让用户编辑行，并且更新数据库中的数据。选中智能标记内的“启用编辑”复选框，即可添加编辑功能。相应地，系统自动生成如下代码：

```
<asp:CommandField ShowSelectButton = "True" ShowEditButton = "True" />
```

（6）使用 TemplateField 模板字段

GridView 是由一组字段(Field)组成的，最简单的字段类型是 BoundField，它仅将数据简单地显示为文本。其他的字段类型使用交互 HTML 元素(alternate HTML elements)来显示数据。例如，CheckBoxField 将被呈现为一个 CheckBox，其选中状态由某特定数据字段的值来决定；ImageField 则将某特定数据字段呈现为一张图片，当然，这个数据字段中应该放的是图片类型的数据。超级链接和按钮的状态取决于使用 HyperLinkField 或 ButtonField 字段类型的数据字段的值。

虽然 CheckBoxField、ImageField、HyperLinkField 和 ButtonField 考虑到了数据的交互视图，但它们仍然有一些相关的格式化的限制。CheckBoxField 只可以显示为一个单个的 CheckBox，一个 ImageField 则只可以显示为一张图片。如果某个字段要显示一些文本、复选框、图片还有一些其他基于不同数据的内容时，我们要做什么？或者说，如果我们需要使用除了 CheckBox、Image、HyperLink 以及 Button 之外的 Web 控件来显示数据时，我们该怎么办？此外，BoundField 只能显示一个单独的数据字段。如果我们想要在一个 GridView 列中显示两个或者更多的数据字段的值的时候该怎么办呢？

为了满足这样一种复杂的情况，GridView 提供了使用模板来进行呈现的 TemplateField。模板可以包括静态的 HTML、Web 控件以及数据绑定的代码。此外，TemplateField 还拥有各种可以用于不同情况的页面呈现的模板。比如说，ItemTemplate 是默认的用于呈现每行中的单元格的，而 EditItemTemplate 则用于编辑数据时的自定义界面。典型的包含 ItemTemplate 模板的 GridView 控件代码如下所示：

```
<asp:GridView ID = "GridViewBook" runat = "server" AllowPaging = "True" AllowSorting = "True"
                AutoGenerateColumns = "False" CellPadding = "4" DataKeyNames = "title_id"
                DataSourceID = "SourceBooks"   ForeColor = "#333333" GridLines = "None"
                Width = "440px">
    <FooterStyle BackColor = "#990000" Font - Bold = "True" ForeColor = "White" />
    <Columns>
      <asp:BoundField DataField = "title" HeaderText = "title" SortExpression = "title" />
      <asp:BoundField DataField = "type" HeaderText = "type" SortExpression = "type" />
      <asp:TemplateField HeaderText = "image">
          <EditItemTemplate>
              <asp:TextBox ID = "TextBox1" runat = "server" Text = '<%# Eval("image") %>'/>
          </EditItemTemplate>
          <ItemTemplate>
             <asp:Image ID = "Image2" runat = "server"
    ImageUrl = '<%# Eval("image", "~\\images\\bookImages\\{0}") %>'
    Height = "50%" Width = "50%" />
           </ItemTemplate>
       </asp:TemplateField>
       <asp:CommandField ShowSelectButton = "True" >
```

```
            <ItemStyle Width="80px" />
        </asp:CommandField>
    </Columns>
    <RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
    <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True" ForeColor="Navy" />
    <PagerStyle BackColor="#FFCC66" ForeColor="#333333" HorizontalAlign="Center" />
    <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
<AlternatingRowStyle BackColor="White" />
</asp:GridView>
```

4.4 项目实施

项目实施过程分解为4个任务，每个任务通过若干步完成。在接下来的内容里，将介绍各任务的实现过程。

4.4.1 任务4-1 采用DataList控件实现图书信息浏览功能

1. 任务介绍

网上书店网站需要一些页面来介绍图书信息。本任务就是创建这样的网页，在该网页上，用户可以浏览所有的图书信息，包括书名、作者、价格、内容提要等，这些信息以表格的形式展现，既包含文本，也包含价格等格式化数字。

2. 任务分析

图书信息存放在数据库NetBook中，数据源控件SqlDataSource负责查询和操作数据库，查询的结果就用显示控件来展示。常用的显示控件有GridView、DataList、DetailsView等，在本任务中用DataList控件来实现。因此，本任务的核心工作就是设置SqlDataSource和DataList控件，实现用SqlDataSource控件读取数据，用DataList控件显示数据。

(1) 创建显示图书信息的Web页面

首先，创建一个存放有关产品信息Web页的文件夹products，这也对应着系统设计中的一个功能模块。在解决方案窗口的网站名称上右击，在弹出的菜单中选择“新建文件夹”，取名为products。在文件夹products上右击，在弹出的菜单中选择“添加新项”，系统弹出“添加新项”对话框，如图4-6所示。

选择新建Web窗体，并将它命名为BookInformation.aspx，确认选中的语言是Visual C#，并且选中了“选择母版页”复选框，然后单击“添加”按钮，转到“选择母版页”对话框，如图4-7所示。

选中了母版页MasterPage.master后，单击“确定”按钮，生成相应的Web窗体。在设计模式下从工具箱的“HTML”控件面板上拖放一个表格Table，默认为三行三列，并将它宽度设为“100%”。从工具箱的“数据”控件面板上分别拖放一个DataList控件和一个SqlDataSource控件到Web窗体上，如图4-8所示。

至此，我们就创建了一个准备用来显示图书信息的Web页面。



图 4-6　“添加新项”对话框



图 4-7　“选择母版页”对话框

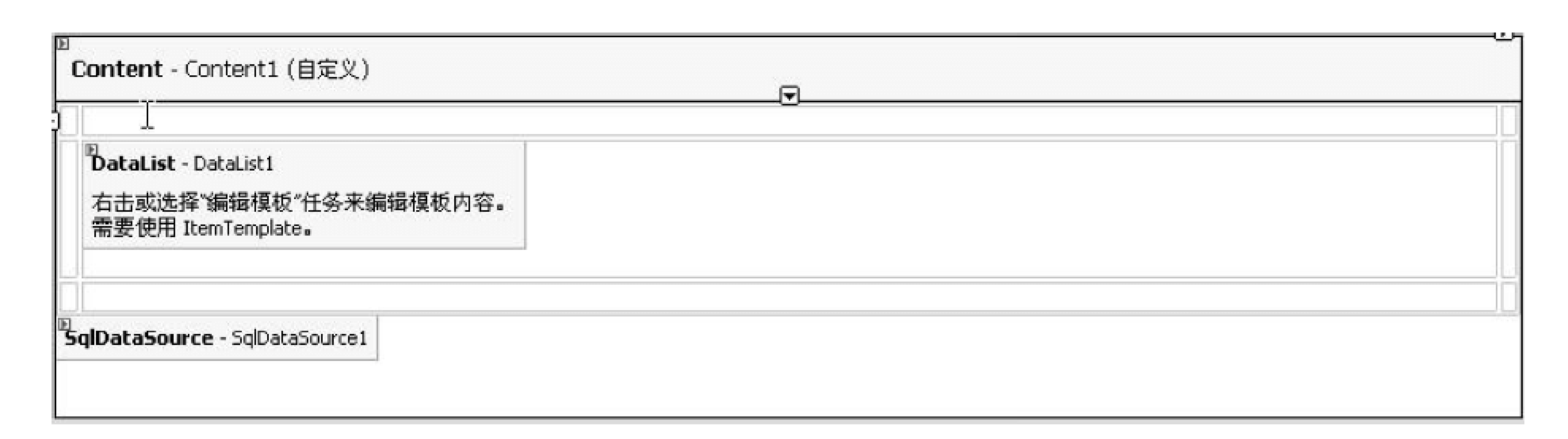


图 4-8　采用 DataList 显示图书信息的 Web 页面

（2）设置 SqlDataSource 数据源

本步骤的目的是配置数据源控件，访问数据库以取得图书信息。首先单击 SqlDataSource 控件的智能按钮，在弹出的菜单中选择“配置数据源…”，打开“配置数据源”对话框，如图 4-9 所示。



图 4-9 “配置数据源”对话框

单击界面上的“新建连接…”按钮，弹出“添加连接”对话框，如图 4-10 所示。

因为我们要连接的是 SQL Server 数据库实例，而不是单一的数据库文件，所以要单击“更改…”按钮，弹出“更改数据源”对话框，如图 4-11 所示。







图 4-10 “添加连接”对话框　　　图 4-11 “更改数据源”对话框

选中数据源列表中的“Microsoft SQL Server”选项，确认数据源提供程序是“用于 SQL Server 的 .NET Framework”，单击“确定”按钮，再回到“添加连接”对话框，如图 4-12 所示。

在“服务器名”下拉列表中选择当前的数据库服务器，也可输入“localhost”，紧接着在“选择或输入一个数据库名”下拉列表中选中要使用的数据库“NetBook”，单击“测试连接”按钮，弹出测试结果对话框，如图 4-13 所示。

如果一切顺利，会报告测试连接成功，再单击“确定”按钮，返回“配置数据源”对话框的“选择您的数据连接”步骤，如图 4-14 所示。



图 4-12　“添加连接”对话框



图 4-13　测试结果对话框



图 4-14　“配置数据源”对话框的“选择您的数据连接”步骤

单击“连接字符串”旁边的“+”按钮，会弹出具体的连接字符串，确认无误后，单击“下一步”按钮，进到“配置数据源”对话框的“将连接字符串保存到应用程序配置文件中”步骤，如图 4-15 所示。

在“是否将连接保存到应用程序配置文件中?”问题中勾选“是，将此连接另存为”复选框，在下面的文本框中输入“NetBookConnectionString”，作为我们保存在配置文件中的连



图 4-15 “配置数据源”对话框的“将连接字符串保存到应用程序配置文件中”步骤

接字符串的名称。网站的 web. config 文件中将自动插入如下的内容：

```
< add name = "NetBookConnectionString" connectionString = "Data Source = localhost; Initial
Catalog = netbook; Integrated Security = True"    providerName = "System. Data. SqlClient" />
```

单击“下一步”按钮，进到“配置数据源”对话框的“配置 Select 语句”步骤，如图 4-16 所示。



图 4-16 “配置数据源”对话框的“配置 Select 语句”步骤

选中“指定来自表或视图的列”单选按钮，“名称”下拉列表列出了当前数据库中可用的表或视图，从中选择存储图书信息的表 t_book，下面的“列”选项区列出了 t_book 表中的全

部字段，按图 4-16 所示勾选所需的字段，然后单击“下一步”按钮，进到“配置数据源”对话框的“测试查询”步骤，如图 4-17 所示。



图 4-17　“配置数据源”对话框的“测试查询”步骤

在图 4-17 中，“SELECT 语句”下方列出了当前系统根据设置自动生成的 SELECT 查询语句。为了测试它的效果，单击“测试查询”按钮，系统以表格形式返回了 SELECT 语句的查询结果，这正是我们配置 SqlDataSource 控件的预期目的，至此，SqlDataSource 控件就配置完成了。

(3) 设置 DataList 控件

配置好 SqlDataSource 控件后，我们就要设置 DataList 控件来显示查询到的数据。首先，单击 DataList 控件的智能按钮，弹出“DataList 任务”菜单，如图 4-18 所示。



图 4-18　“DataList 任务”菜单

“选择数据源”下拉列表列出了当前可用的全部数据源，选择先前配置好的“SqlDataSource1”。由于选择了数据源，系统根据所选数据源控件的配置情况，设置 DataList 控件的默认显示内容，如图 4-19 所示。

DataList 控件的默认显示内容列出了数据源控件 SqlDataSource 的查询命令所返回的全部数据，以数据库中表的字段名作为显示字段名，并且格式采用简单流格式，看起来并不美观。为美化其外观，单击智能菜单上的“自动套用格式”项，选择“彩色型”，再单击智能菜



图 4-19　DataList 控件的默认显示内容

单上的“编辑模板”项，弹出显示模板 ItemTemplate，如图 4-20 所示。

根据实际需要，我们对显示模板 ItemTemplate 中的内容作如图 4-21 所示的调整。

单击 pubdateLabel 的智能按钮，单击菜单项“编辑 DataBindings…”，在弹出的对话框中将它的 Text 格式选为“长日期-{0:D}”。

单击 priceLabel 的智能按钮，单击菜单项“编辑 DataBindings…”，在弹出的对话框中将它的 Text 格式选为“货币-{0:C}”。

运行后效果如图 4-22 所示。



图 4-20　显示模板 ItemTemplate 中的默认布局

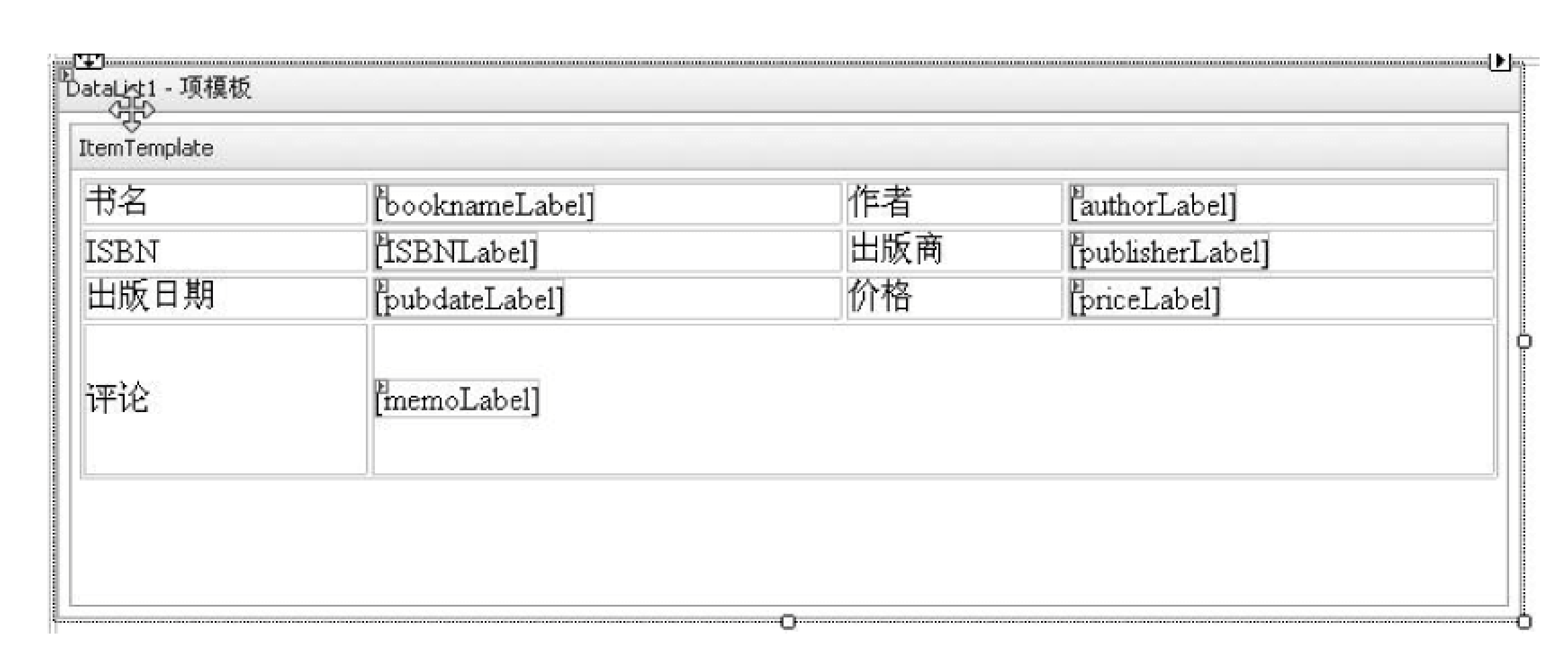


图 4-21　显示模板 ItemTemplate 中调整后的布局

书名	明朝那些事儿	作者	当年明月
ISBN	9787801655998	出版商	中国海关
出版日期	2002年3月1日	价格	￥23.00
评论	明朝历史上的最后一位皇帝，自来有许多传奇。关于崇祯帝究竟是一个昏庸无能的皇帝，还是一个力...		
书名	牧羊少年奇幻之旅	作者	（巴西）柯艾略
ISBN	9787801634583	出版商	南海
出版日期	2003年4月1日	价格	￥25.00
评论	迄今唯一一部语种超过《圣经》的书　在巴西，按知名度他与上帝、足球并列　在美国，他是唯...		
书名	村级干部	作者	贺享雍
ISBN	9787333633283	出版商	上海
出版日期	2002年4月8日	价格	￥16.00
评论	一座百年老屋，见证几代恩仇；两个坚强女性，撬动僻静山村。跌宕起伏的故事情节，变化多端的人物命...		

图 4-22　DataList 控件运行效果

3. 任务完成总结

实现图书信息浏览功能的关键点是将 DataList 控件与数据源控件 SqlDataSource 绑定，由 SqlDataSource 控件连接到数据库读取信息，用 DataList 控件来显示这些图书信息。设置 SqlDataSource 控件主要就是设置它的 SelectCommand、UpdateCommand、InsertCommand 和 DeleteCommand，我们可以用向导工具实现。设置 DataList 控件主要就是设置 ItemTemplate 中各字段的排列形式，使图书信息能按比较有序、美观的方式展示。

4. 课堂训练与知识拓展

图书信息浏览是读者经常访问的页面，所以它的界面设计非常重要，强调美观有序，能吸引读者。DataList 控件提供了多种模板，可以布置 HTML 元素，并且通过采用不同字体、颜色和背景图案，做到不仅内容正确，而且界面活泼，赏心悦目。

4.4.2　任务 4-2　采用 GridView 控件实现图书信息浏览功能

1. 任务介绍

网上书店网站需要一些页面来介绍图书信息，这些页面可能采用不同的技术来实现。本任务就是创建一个这样的网页，用户可以浏览所有的图书信息，包括书名、作者、价格、内容提要等，这些信息以表格的形式展现，既包含文本，也包含图像。

2. 任务分析

图书信息存放在数据库 NetBook 中，数据源控件 SqlDataSource 负责查询和操作数据库，查询的结果就用显示控件来展示。常用的显示控件有 GridView、DataList、DetailsView

等，在本任务中用GridView控件来实现。因此，本任务的核心工作就是设置SqlDataSource和GridView控件，实现用SqlDataSource控件读取数据，用DataList控件显示数据。

（1）创建显示图书信息的Web页面

采用和4.4.1小节任务4-1中步骤1相同的方法，创建一个Web页面BooksInfo.aspx，在页面中拖放入一个GridView控件和一个SqlDataSource控件，如图4-23所示。

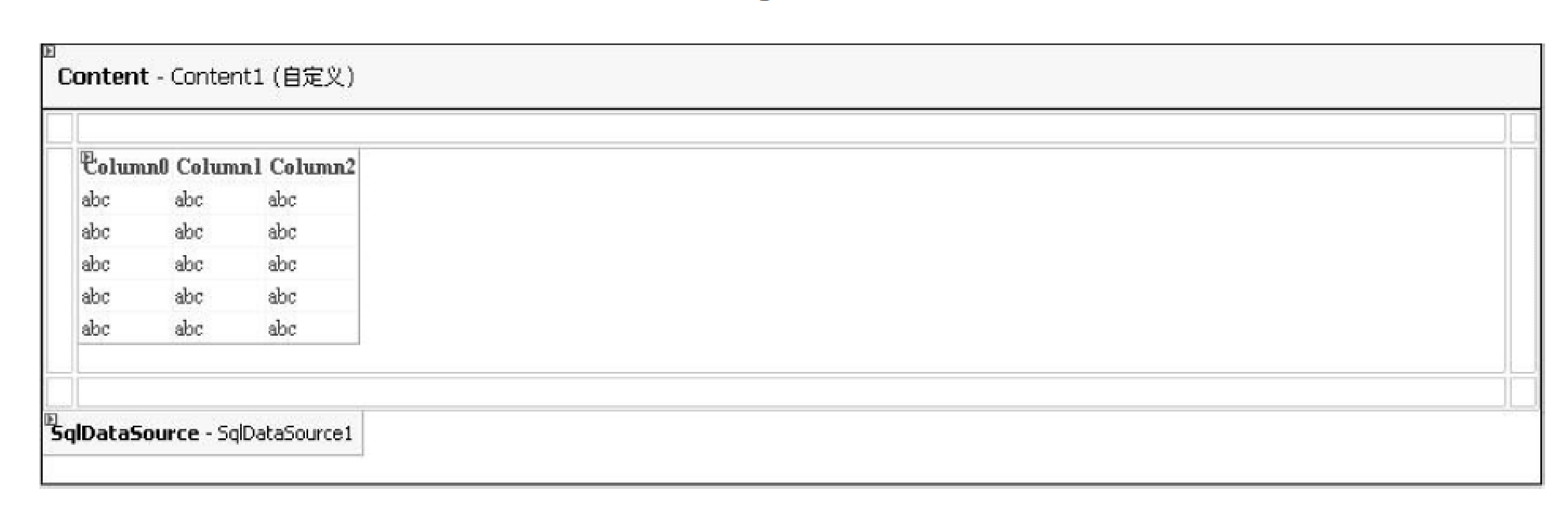


图4-23　采用GridView控件的页面

（2）设置SqlDataSource数据源

采用和4.4.1小节任务4-1中步骤2类似的方法来设置SqlDataSource数据源。在“配置数据源”对话框的“选择您的数据连接”步骤中，可以从下拉列表中选中NetBookConnectionString项，避免从开始连接到数据库，如图4-24所示。



图4-24　选用“NetBookConnectionString”数据连接

在“配置数据源”对话框的“配置Select语句”步骤，比上次多选一个字段picture，如图4-25所示。

依次单击“下一步”按钮完成数据源SqlDataSource的设置。

（3）设置GridView控件

配置好SqlDataSource控件后，就要设置GridView控件来显示查询到的数据。首先，单击GridView控件的智能按钮，弹出“GridView任务”菜单，如图4-26所示。



图 4-25 “配置 Select 语句”步骤

“选择数据源”下拉列表列出了当前可用的全部数据源，选择先前配置好的 SqlDataSource1，如图 4-27 所示。

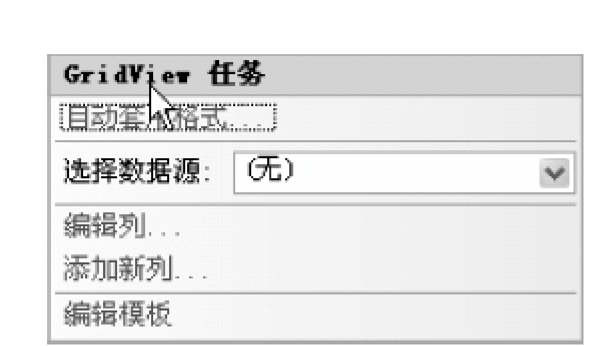


图 4-26 “GridView 任务”菜单



图 4-27 更新后的“GridView 任务”菜单

由于选择了数据源，系统根据我们所选数据源控件的配置情况，增添了可选功能，更新了“GridView 任务”菜单。下面我们要设置 GridView 控件的显示外观，先单击“编辑列…”菜单项，弹出“字段”对话框，如图 4-28 所示。

在对话框的左上角列出了所有可用的字段，左下角列出了所有选定的字段，右侧属性表列出了当前选中字段的所有属性。根据实际情况，我们不需向用户显示图书信息的 ISBN 号，所以在左下角的列表框中选定“ISBN”字段，单击列表框右侧的“×”按钮删掉这个字段。同样，把 publisher 字段也删掉。在“选定的字段”列表框选中 bookname，单击右边向上箭头，可以将它的排序位置向上移，经过类似的调整，我们可以使显示的字段次序符合我们的要求。

接下来的工作是设置各字段的属性，选中“bookname”字段，它默认的标题名称和字段名一样，也是“bookname”，为了将它改为中文，我们选中“HeaderText”属性，输入“书名”，然后依次将其他字段的标题名称改为“作者”、“价格”、“出版日期”、“备注”和“封面”。有些字



图 4-28 “字段”对话框

段的显示格式还需要设置一下，如选中“Price”字段的 DataFormatString 属性，输入格式字符串“{0:C}”，表示它用货币形式显示，为了显示这些设定格式，将 HtmlEncode 属性设置为 False，将 ApplyFormatInEditMode 属性设置为 True，最后单击“确定”按钮，关闭对话框。

为了设置 GridView 控件的显示外观，单击“GridView 任务”智能菜单的“自动套用格式…”菜单项，弹出“自动套用格式”对话框，如图 4-29 所示。

对话框的左侧列出了所有可选的配色方案，当单击它们时，右侧会相应显示对应的配色外观，我们选中“彩色型”，单击“确定”按钮，关闭对话框。

经过以上的配置，Web 页面上 GridView 控件就具备了如图 4-30 所示的外观。



图 4-29 “自动套用格式”对话框

书名	作者	价格	出版日期	评论	封面
abc	abc	￥0.00	2009-8-8	abc	abc
abc	abc	￥0.10	2009-8-8	abc	abc
abc	abc	￥0.20	2009-8-8	abc	abc
abc	abc	￥0.30	2009-8-8	abc	abc
abc	abc	￥0.40	2009-8-8	abc	abc

图 4-30 Web 页面上的 GridView 控件外观效果

按 F5 键调试程序，将显示如图 4-31 所示的运行时结果。

从图 4-31 可以看到，封面栏显示的是封面图像文件的路径，而不是实际图像，为此我们要修改显示控件。默认状态下字段内容采用标签控件显示，而显示图像需用 Image 控件。在“GridView 任务”智能菜单上单击“编辑列…”菜单项，在弹出的“字段”对话框中选中“封面”字段，如图 4-32 所示。

书名	作者	价格	出版日期	评论	封面
明朝那些事儿	当年明月	￥23.00	2002-3-1	明朝历史上的最后一位皇帝，自来有许多传奇。关于崇祯帝究竟是一个昏庸无能的皇帝，还是一个力...	image/1.jpg
牧羊少年奇幻之旅	（巴西）柯艾略	￥25.00	2003-4-1	迄今唯一一部语种超过《圣经》的书　在巴西，按知名度他与上帝、足球并列　在美国，他是唯...	image/2.jpg
村级干部	贺享雍	￥16.00	2002-4-8	一座百年老屋，见证几代恩仇；两个坚强女性，撬动僻静山村。跌宕起伏的故事情节，变化多端的人物命...	image/3.jpg
贫民窟的百万富翁	（印）斯瓦鲁普	￥34.00	2002-2-12	十八岁的酒吧服务员罗摩，生活在孟买的贫民窟里。他参加了一个名为《谁将赢得十个亿》的电视知识问...	image/4.jpg
致命的温柔	艾米	￥35.00	2003-5-8	一次偶然的接机，Carol和Jason一见钟情，飞蛾扑火般地爱上了对方，他们在迷恋中伤害对方，又在伤害中更加怜惜彼此。坚强执着的Carol，却因为父亲当年的背弃而不敢相信真爱；温柔体贴的Jason，因为一个女孩曾为他自杀，而患上了“重症爱无力”。两人互相吸引却又互相逃避，明知道自己	image/5.jpg
项羽	李可	￥17.50	2002-12-5	本书是一部历史学家撰写的英雄传奇，以项羽的崛起和楚汉相争为线索，展开了战国末年到秦亡汉兴群雄争霸的一段波澜壮阔的历史。	image/6.jpg

图 4-31　运行时 GridView 控件外观效果



图 4-32　“字段”对话框

选中“封面”字段后，单击“将此字段转换为 TemplateField”链接，再在“GridView 任务”智能菜单上单击“编辑模板”，弹出如图 4-33 所示编辑模板界面。

在模板上看到一个默认的 Label 控件，它的 Text 属性绑定到 picture 字段，所以在运行时我们只看到封面图像的路径。删掉这个 Label 控件，添加一个 Image 控件，如图 4-34 所示。



图 4-33　编辑模板界面



图 4-34　在 picture 模板上添加 Image 控件

单击 Image 控件的智能按钮，在弹出的任务菜单中单击“编辑 DataBindings…”菜单项，弹出如图 4-35 所示对话框。

在左侧“可绑定属性”栏中，选中“ImageUrl”，在右侧“绑定到”下拉列表中选中 picture 字段，由于 picture 字段的内容已包含目录名，所以在“格式”列表框中输入“~//{0}”，这些设置表示 Image 控件的图像路径绑定到 picture 字段。最后单击“确定”按钮结束设置。

由于查询得到的数据较多，一次显示出来就使得整个页面很长，为了改善最终效果，在“GridView 任务”智能菜单上勾选“启用分页”和“启用排序”复选框，如图 4-36 所示。

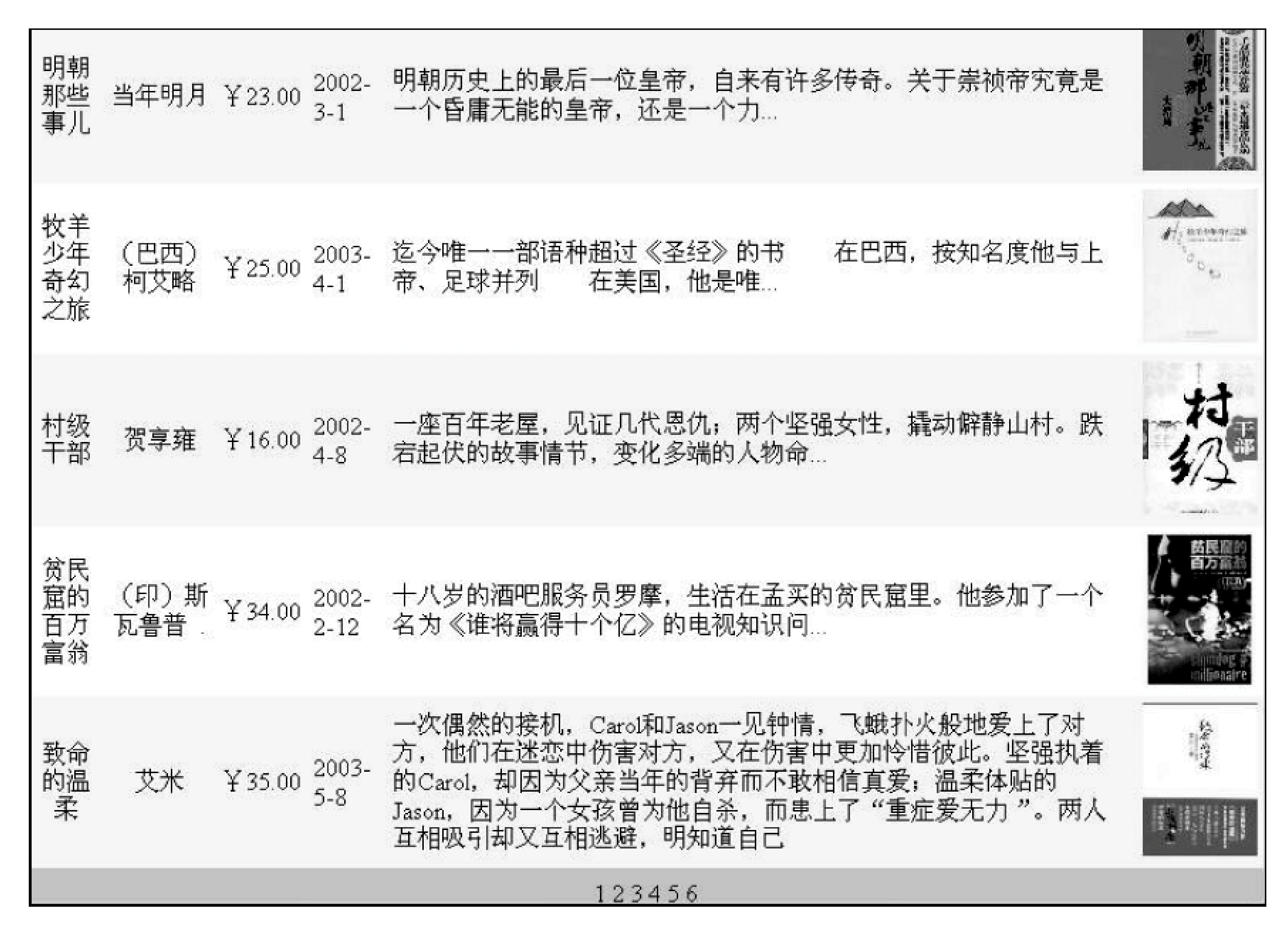


图 4-35 Image1DataBindings 对话框



图 4-36 “启用分页”和“启用排序”复选框

将 GridView 控件的 PageSize 属性设为 5，表示每次显示 5 条记录，按 F5 键调试程序，将显示如图 4-37 所示的运行时最终效果。



明朝那些事儿	当年明月	￥23.00	2002-3-1	明朝历史上的最后一位皇帝，自来有许多传奇。关于崇祯帝究竟是一个昏庸无能的皇帝，还是一个力...	
牧羊少年奇幻之旅	（巴西）柯艾略	￥25.00	2003-4-1	迄今唯一一部语种超过《圣经》的书 在巴西，按知名度他与上帝、足球并列 在美国，他是唯...	
村级干部	贺享雍	￥16.00	2002-4-8	一座百年老屋，见证几代恩仇；两个坚强女性，撬动僻静山村。跌宕起伏的故事情节，变化多端的人物命...	
贫民窟的百万富翁	（印）斯瓦鲁普	￥34.00	2002-2-12	十八岁的酒吧服务员罗摩，生活在孟买的贫民窟里。他参加了一个名为《谁将赢得十个亿》的电视知识问...	
致命的温柔	艾米	￥35.00	2003-5-8	一次偶然的接机，Carol和Jason一见钟情，飞蛾扑火般地爱上了对方，他们在迷恋中伤害对方，又在伤害中更加怜惜彼此。坚强执着的Carol，却因为父亲当年的背弃而不敢相信真爱；温柔体贴的Jason，因为一个女孩曾为他自杀，而患上了“重症爱无力”。两人互相吸引却又互相逃避，明知道自己	

1 2 3 4 5 6

图 4-37 GridView 控件运行时最终效果

3. 任务完成总结

实现图书信息浏览的关键点是将 GridView 控件与数据源控件 SqlDataSource 绑定，由 SqlDataSource 控件连接到数据库读取信息，用 GridView 控件来显示这些图书信息。设置 SqlDataSource 控件主要就是设置它的 SelectCommand、UpdateCommand、InsertCommand 和 DeleteCommand，我们可以用向导工具实现。设置 GridView 控件主要就是设置控件的外观和各字段的属性、排列形式等，使整个表格能按比较有序、美观的方式展示。如要某个字段要显示图案，则必须将它转化为模板字段，再在其中加入一个 Image 控件，绑定它的 ImageUrl 属性，显示指定的图像。

4. 课堂训练与知识拓展

图书信息浏览是读者经常访问的页面，它的界面非常强调美观有序，以吸引读者。GridView 控件提供了丰富而强大的配置功能，可以非常方便地获得一般效果。如果需要特殊的内容，可以将某个字段转成模板字段，在里面可以设置 HTML 元素，通过插入新的内容、采用不同的字体颜色和背景图案，做到不仅内容正确，而且界面活泼，赏心悦目。

4.4.3 任务 4-3 实现图书信息分类浏览功能

1. 任务介绍

网上书店网站需要一些页面来介绍图书信息，由于图书数量一般比较大，所以需要按某种类别进行分类，例如小说、诗歌、IT 教材等。本任务将创建一个这样的网页，用户可以先选中某个类别，然后浏览这个类别所有的图书信息。

2. 任务分析

图书类别信息存放在数据库 NetBook 中的 t_category 表，图书的信息存放在 t_book 表，本任务的工作流程就是先从 t_category 表读取数据在下拉列表中展示，用户选中类别后将它作为参数访问 t_book 表，读取全部属于这个类别的图书信息并在 GridView 控件中显示出来。因此，本任务的核心工作是设置两个 SqlDataSource 控件，分别实现读取类别和图书数据，然后将其用不同的控件显示出来。访问 t_book 表的 SqlDataSource 控件中 SelectCommand 带有一个参数，由下拉列表控件提供。

(1) 创建分类显示图书信息的 Web 页面

首先，创建一个分类显示图书信息的 Web 页面，按照与 4.4.2 小节任务 2 中步骤 1 类似的方法新建一个 Web 窗体，并将它命名为 BooksByType.aspx，从工具箱的“数据”控件面板上拖放两个 SqlDataSource 控件到表格的下方，分别命名为“TypeSource”和“BooksSource”，再拖放一个 GridView 控件到中间位置；从工具箱的“标准”控件面板上拖放两个 Label 控件到左侧位置，将它们的 Text 属性分别设为“请选择图书类型：”和“图书信息：”，再拖放一个下拉列表框控件到上部位置。

至此，就创建了一个准备分类显示图书信息的 Web 页面，如图 4-38 所示。

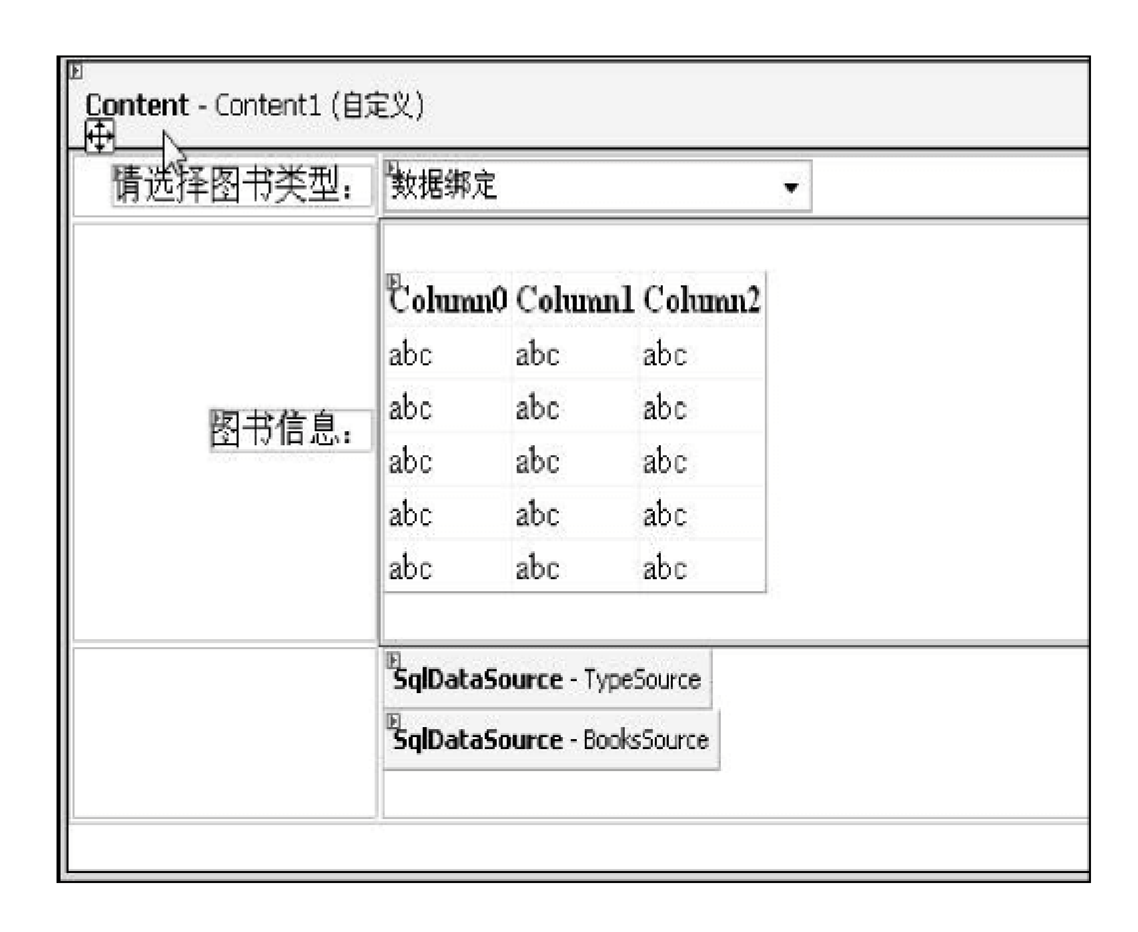


图 4-38 分类显示图书信息的 Web 页面

(2) 设置显示图书类别的数据源

本步骤的目的是配置数据源控件 TypeSource,使它能访问数据库并查询图书类别信息。

和 4.4.2 小节任务 4-2 中的步骤 2 类似,首先单击控件 TypeSource 的智能按钮,在弹出的菜单中选择“配置数据源…”,打开“配置数据源”对话框,在可选数据源下拉列表中选中先前建立的连接“NetBookConnectionString”,单击“下一步”按钮,进入“配置数据源”对话框的“配置 Select 语句”步骤,如图 4-39 所示。

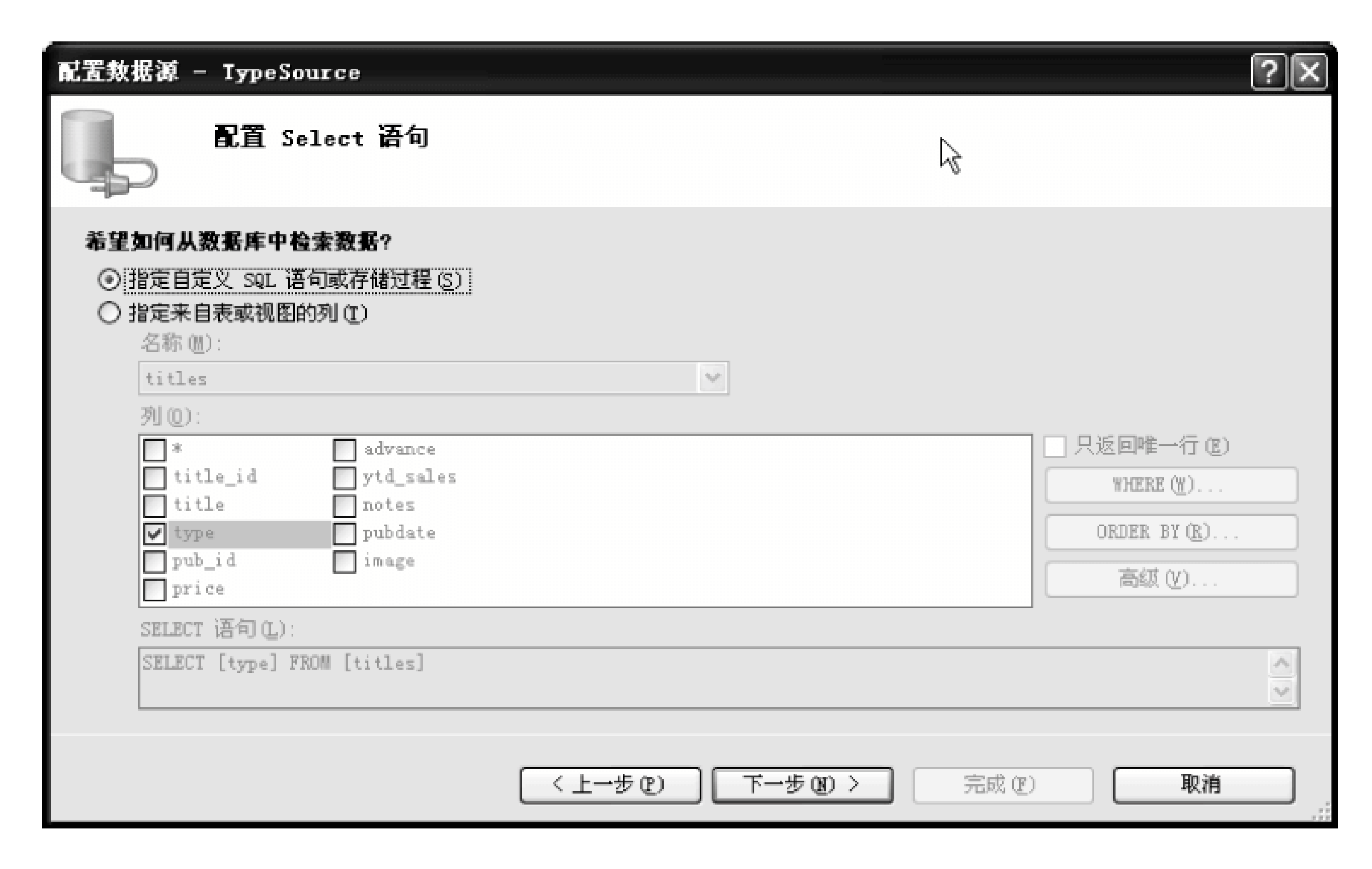


图 4-39 数据源控件 TypeSource 的“配置 Select 语句”步骤

因为查询图书类别信息并不是简单地返回一个表或视图的字段,所以在“配置 Select 语句”步骤中选中“指定自定义 SQL 语句或存储过程”单选按钮,单击“下一步”按钮,进到“配置数据源”对话框的“定义自定义 SQL 语句或存储过程”步骤,如图 4-40 所示。

图 4-40 中有 4 个选项卡,可以分别为 SELECT、UPDATE、INSERT 和 DELETE 操作定义 SQL 语句或存储过程。我们可以在 SQL 语句文本框直接输入自定义语句,也可以使



图 4-40 “配置数据源”对话框的“定义自定义 SQL 语句或存储过程”步骤

用内建的向导工具来图形化地生成自定义语句，所以单击“查询生成器…”按钮，进到查询向导，首先弹出“添加表”对话框，如图 4-41 所示。

在这个对话框中列出了所有可选的表、视图和函数，选择“t_category”表，单击“添加”按钮，再单击“关闭”按钮，转到“查询生成器”对话框，如图 4-42 所示。

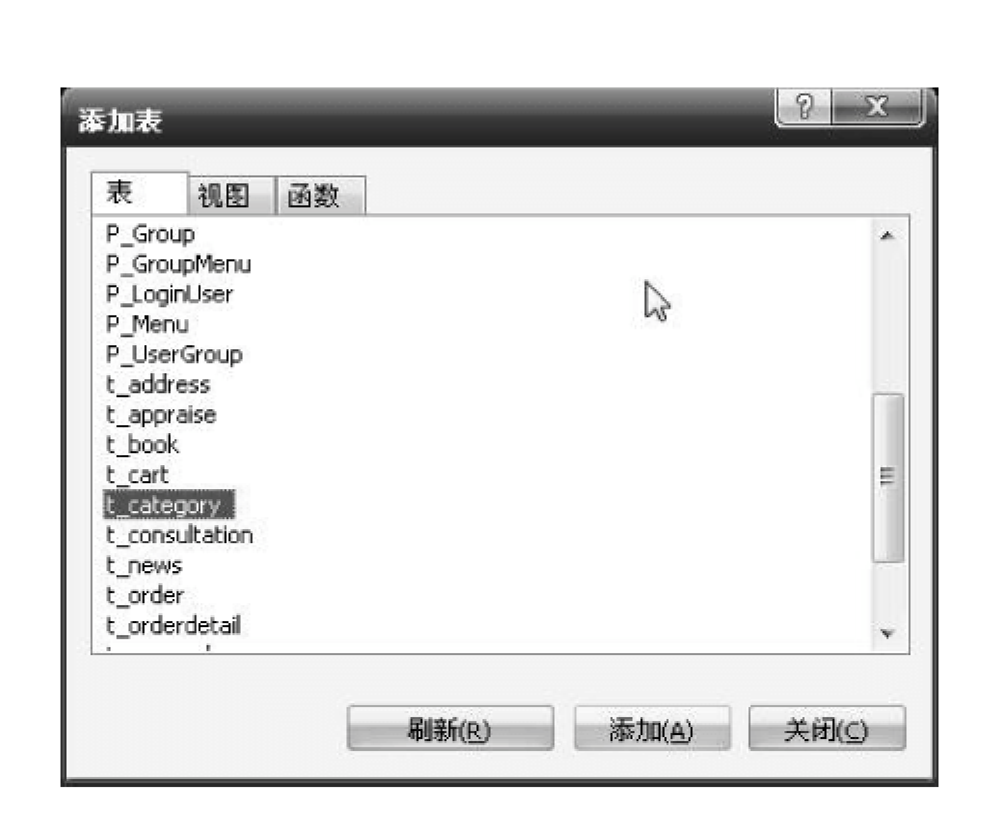



图 4-41 “添加表”对话框



图 4-42 “查询生成器”对话框

在 t_category 表的字段中勾选 CategoryId 和 CategoryName，在对话框的中间显示出了系统自动生成的 SELECT 语句。为了测试 SELECT 语句的查询效果，单击“执行查询”按钮，在对话框的下部列出了查询返回的结果，完全符合设置预期，单击“确定”按钮，关闭“查询生成器”，返回“配置数据源”对话框，依次单击“下一步”按钮，直到完成。

(3) 设置显示图书类别的数据显示控件

配置好显示图书类别的数据源控件 typeSource 后，我们就要设置下拉列表控件

DropDownList1 来显示查询到的图书类别。首先,单击下拉列表框控件的智能按钮,弹出"DropDownList 任务"菜单,如图 4-43 所示。

在这个"DropDownList 任务"菜单中,确认选中了"启用 AutoPostBack"复选框,单击"选择数据源…"菜单项,转到"数据源配置向导"对话框,如图 4-44 所示。



图 4-43 "DropDownList 任务"菜单



图 4-44 "数据源配置向导"对话框

在"选择数据源"下拉列表中,选中在上一步骤配置好的"TypeSource"数据源,在"选择要在 DropDownList 中显示的数据字段"下拉列表中选中字段"CategoryName",在"为 DropDownList 的值选择数据字段"下拉列表中选中字段"CategoryId",然后单击"确定"按钮,完成本步骤的设置。

按 F5 键调试程序,将显示如图 4-45 所示的运行时图书类别效果。

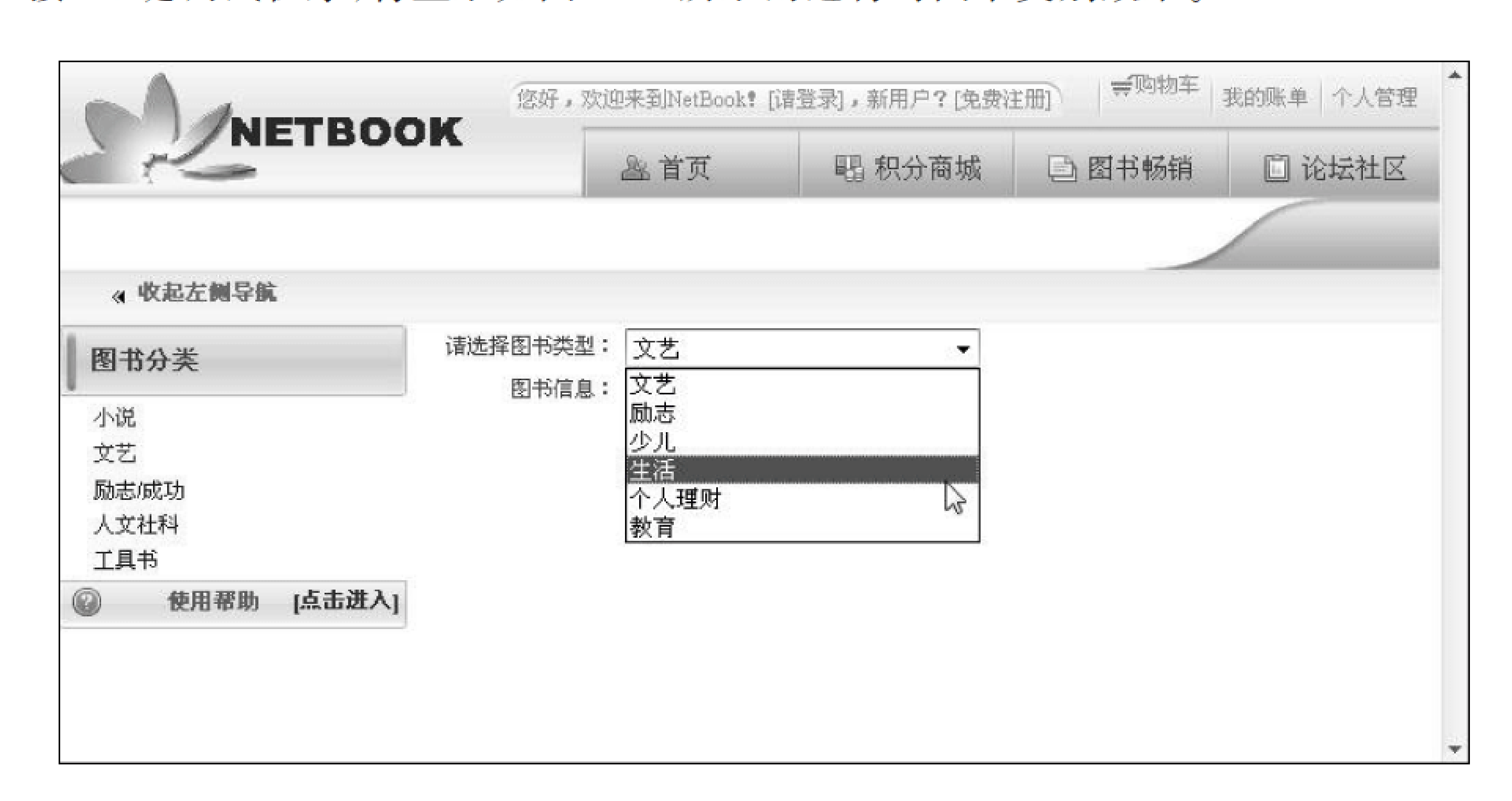


图 4-45 运行时图书类别效果

(4) 设置显示特定类别图书信息的数据源

本步骤的目的是配置数据源控件,访问数据库以取得特定类别的图书信息。首先单击控件 BooksSource 的智能按钮,在弹出的菜单中选择"配置数据源…",打开"配置数据源"对

话框，在可选数据源下拉列表中选中先前建立的连接“NetBookConnectionString”，单击“下一步”按钮，进到“配置数据源”对话框的“配置 Select 语句”步骤，如图 4-46 所示。



图 4-46　“配置数据源 BooksSource”的“配置 Select 语句”步骤

按照图 4-46 所示勾选所需的字段后，单击“WHERE…”按钮，进到“添加 WHERE 子句”对话框，如图 4-47 所示。

图 4-47　“添加 WHERE 子句”对话框

在“列”下拉列表中选择“CategoryId”字段；在“运算符”下拉列表中选择“＝”；在“源”下拉列表中选择“Control”，表示 SQL 参数将由控件来提供；在“控件 ID”下拉列表中选择显示图书类别信息的 DropDownList1 选项。单击“添加”按钮，系统在“WHERE 子句”文本框中生成相应的 SQL 表达式，如图 4-48 所示。

整个 SELECT 语句将是：



图 4-48 自动生成的 SQL 表达式

```
SELECT [author], [CategoryId], [picture], [memo], [price], [bookname] FROM [t_book]
WHERE ([CategoryId] = @CategoryId)
```

其中，参数@CategoryId 将由下拉列表"DropDownList1"的选中值提供。单击"确定"按钮，返回"配置数据源"对话框，依次单击"下一步"按钮，直到完成。整个数据源控件"BooksSource"的源代码如下所示：

```
<asp:SqlDataSource ID = "BooksSource" runat = "server"
        ConnectionString = "<% $ ConnectionStrings:NetBookConnectionString %>"
        SelectCommand = "SELECT [author], [CategoryId], [picture], [memo], [price], [bookname]
                    FROM [t_book] WHERE ([CategoryId] = @CategoryId)">
    <SelectParameters>
        <asp:ControlParameter ControlID = "DropDownList1" Name = "CategoryId"
                    PropertyName = "SelectedValue"  Type = "Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

(5) 设置显示图书信息的数据显示控件

配置好显示特定类别图书信息的数据源控件 BooksSource 后，就要设置 GridView 控件来显示查询到的特定类别图书信息。整个设置步骤与 4.4.2 小节任务 4-2 的步骤 3 非常类似，数据源选择"BooksSource"，结果如图 4-49 所示。

书名	作者	价格	封面	介绍
abc	abc	￥0.00		abc
abc	abc	￥0.10		abc
abc	abc	￥0.20		abc
abc	abc	￥0.30		abc
abc	abc	￥0.40		abc
1 2				

图 4-49 显示图书信息的 GridView 控件

按 F5 键调试程序，将显示如图 4-50 所示的运行时效果。



图 4-50　运行时分类显示图书信息效果

用户在图书类型下拉列表中选中某一类别后，下面的 GridView 控件就会显示这一特定类别所有图书的信息。

3. 任务完成总结

本任务是一个典型的主从数据读取模式任务，图书类别信息作为主数据，图书信息作为从数据，主数据确定后作为参数读取从数据，从而完成整个任务。主数据用一个用户可以选定的控件展示，例如下拉列表和 GridView，这个控件将为从数据的读取提供参数。

4. 课堂训练与知识拓展

图书类别的层次可以有多层，例如小说类别下还可以有古典小说、侦探小说和玄幻小说等，因此主从数据读取模式对应的也可以有多层。实践中用户选中了某个大类后（例如小说），作为参数访问数据库获取小类，用户选中小类后（例如玄幻小说）再作为参数访问数据库获取相应的图书信息，从而完成整个分类信息浏览。

4.4.4　任务 4-4　实现图书信息的搜索功能

1. 任务介绍

有些用户需要查找一些特定图书，查找信息可能是书名、作者、出版日期等，网上书店网站需要一个搜索页面来实现这个功能。本任务就是创建一个这样的网页，用户可以输入书名中的某个单词，然后通过网站找出所有书名中包含这个词的图书，并把它展示出来。

2. 任务分析

SqlDataSource 控件中 SelectCommand 用来配置查询语句，而 SQL 查询语句中的 WHERE 子句可以用 LIKE 关键词执行模糊查询，因此本任务的核心就是把用户提供的单词用到 WHERE 子句进行模糊查询。用户可以在一个文本输入框输入查询单词，而这个文本输入框就作为提供参数的控件。查询结果可以放在任何一个显示控件，本任务使用 GridView 控件来实现。

(1) 创建搜索图书信息的 Web 页面

首先,按照与前面类似的方法新建一个 Web 窗体,并将它命名为 BooksSearching.aspx,从工具箱的数据控件面板上拖放一个 SqlDataSource 控件到表格的下方,命名为"BooksSource",再拖放一个 GridView 控件到中间位置;从工具箱的"标准"控件面板上拖放两个 Label 控件到左侧位置,将它们的 Text 属性分别设为"请输入关键字:"和"查询结果:",再拖放一个文本输入框到上部位置,拖放一个按钮到右上部,将其 Text 属性设为"查阅"。至此,我们就创建了查阅图书信息的 Web 页面,如图 4-51 所示。

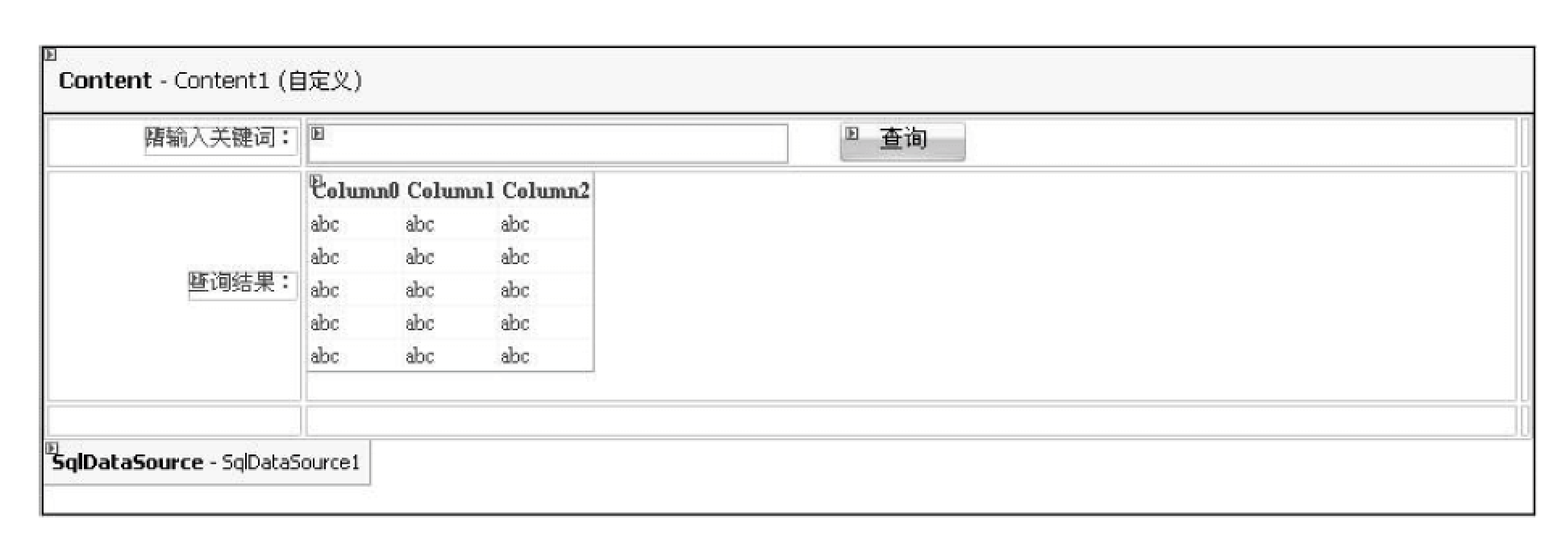


图 4-51 查阅图书信息的 Web 页面

(2) 设置查询图书的数据源

和 4.4.3 小节任务 4-3 中的步骤 4 类似,首先单击控件 TypeSource 的智能按钮,在弹出的菜单中选择"配置数据源…",打开"配置数据源"对话框,在可选数据源下拉列表中选中先前建立的连接"NetBookConnectionString",单击"下一步"按钮,进到"配置数据源"对话框的"配置 Select 语句"步骤,选中 t_book 表,勾选 bookname 等基本信息字段,单击"WHERE…"按钮,进到"添加 WHERE 子句"对话框,如图 4-52 所示。

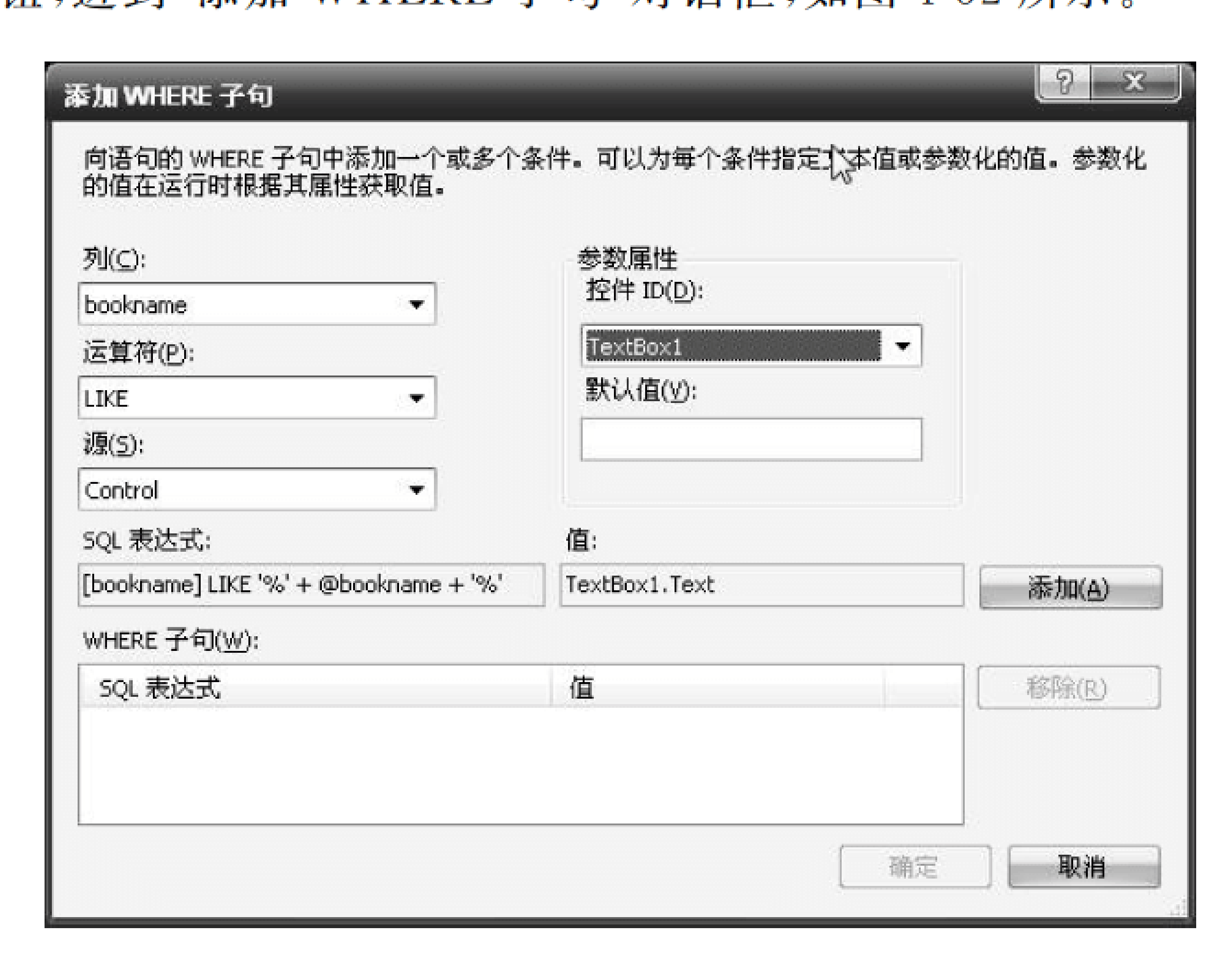


图 4-52 "添加 WHERE 子句"对话框

在"列"下拉列表中选择"bookname"字段;在"运算符"下拉列表中选择"LIKE";在"源"下拉列表中选择"Control",表示 SQL 参数将由控件来提供;在"控件 ID"下拉列表中选择

输入关键字的文本输入框“TextBox1”。单击“添加”按钮，系统在“WHERE 子句”文本框中生成相应的 SQL 表达式，如图 4-53 所示。



图 4-53 带参数的 SQL 表达式

整个 SELECT 语句将是：

```
SELECT [bookname], [author], [price], [picture], [memo] FROM [t_book]
WHERE ([bookname] LIKE '%' + @bookname + '%')
```

其中，参数@bookname 将由文本输入框“TextBox1”提供。单击“确定”按钮，返回“配置数据源”对话框，依次单击“下一步”按钮，直到完成。

(3) 设置显示查询图书结果的显示控件

配置好查询图书的数据源控件 BookSource 后，我们就要设置 GridView 控件来显示查询到的图书信息。整个设置步骤与任务 4-3 中的步骤 7 非常类似，数据源选择“BookSource”，结果如图 4-54 所示。

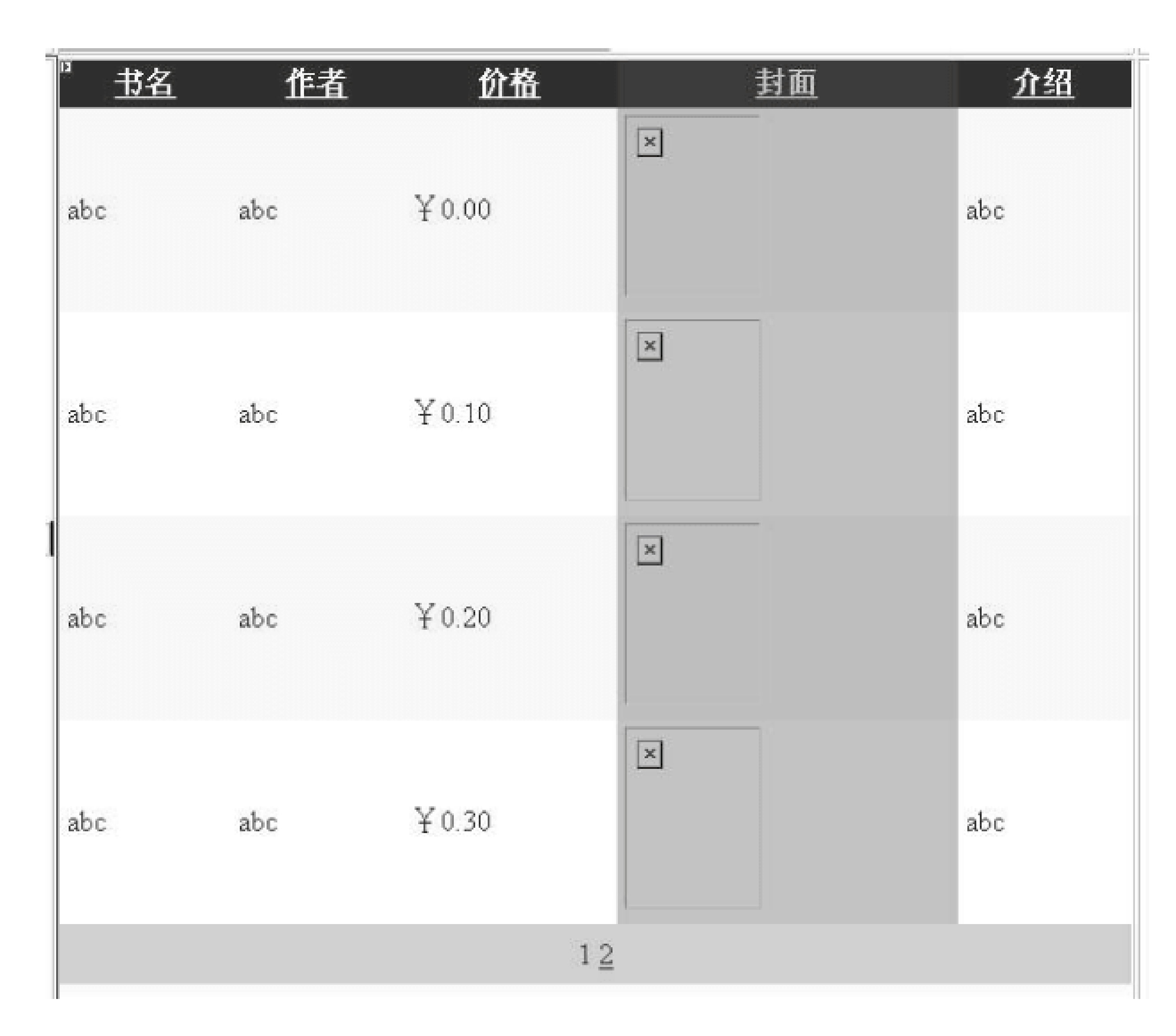


图 4-54 显示查询结果的 GridView 控件

按 F5 键调试程序，将显示如图 4-55 所示的运行时效果。



图 4-55 查询的运行时效果

虽然我们并没有给按钮空间添加响应代码，但因为每次按下按钮后浏览器都会回发请求(post back)，同时将文本框中的关键字也发回服务器，所以经过查询后服务器就会把正确结果发到浏览器。同样的道理，在文本框中按下 Enter 键也会得到相同效果，但注意要将文本框的"AutoPostBack"属性设为 True。

3. 任务完成总结

本任务是一个典型的模糊查询任务，用户提供一个要查询的关键词，然后用这个词进行 SQL 模糊查询，得出结果后再用控件显示出来。此类任务的核心就是提供一些控件让用户输入查询关键字，配置 SqlDataSource 控件的查询语句，使用输入控件的值作为参数进行模糊查询，最后给出查询结果。

4. 课堂训练与知识拓展

用户查询书籍往往是通过多方面的片段信息，例如书名中的一个或几个词、作者、出版阶段等，这些查询约束可以单独进行，也可以组合执行。通过多个控件收集用户的查询约束，把它们作为"与"的关系放入 SQL 查询语句的 WHERE 子句执行模糊查询，得出最后查询结果。

4.5 项目总结

本项目的核心内容是访问数据库，显示查询到的信息。

数据库连接：采用 SqlDataSource 控件。

查询：采用 SqlDataSource 控件的 Select 语句及相关参数。

显示：采用 DataList 和 GridView 等控件。

4.6 项目实训

1. 任务描述

会员在网上书店查阅图书信息，首先通过类别目录选定大类，例如小说，然后查阅这个类别下的所有子类目录，选定某个子类，例如玄幻小说，接着浏览这个子类下的图书信息。有时用户要查阅某类特定图书的信息，例如某个作者的某类作品，并且是最新出版的(也就是最近一年内)，创建一个项目完成这个任务。

2. 任务要求

① 在数据库建立分别反映图书类别与信息的表，并建立良好的关系。

② 创建根据二级目录浏览图书的页面，满足用户分级浏览图书的需求。

③ 创建复合查阅图书信息的页面，使用户可以通过书名中的一个或几个词、作者、出版阶段等信息，获得需要的图书信息。

项目5

实现网上书店图书管理功能

5.1 项目介绍

当创建了网上书店网站后，就可以向用户发布各种图书信息；同时，随着销售的进行和新书的不断发行，还必须不断修改原先的内容，增添新的内容，即要实现网站图书管理的功能。

具体来说，每本图书有书名、作者、价格、简介等信息，随着时间的推移，要能够对这些信息进行修改，例如，如果某种书已售完，并且以后不再进货，那么可以删除这种图书的记录；如果有新的图书发行，则要增添相应的记录供用户浏览。

在这个项目中，要实现对图书信息的修改、删除、增加等功能。

5.2 项目分析

在本项目中将实现的主要功能是修改、增加、删除图书信息，这些信息都保存在 SQL Server 数据库中，因此本项目的核心内容是：

① 连接数据库，设定对数据库中的有关内容进行修改、增加、删除的配置。

② 提供操作界面，方便用户实现对数据库中的有关内容进行操作并检查执行结果。

5.3 相关知识

1. 使用 GridView 控件实现对数据库内容的更新与删除操作

GridView 控件具有一些内置功能，允许用户在不需要编程的情况下编辑或删除记录。可以使用事件和模板来自定义 GridView 控件的编辑或删除功能。

可以通过以下任一方式启用 GridView 控件的内置编辑或删除功能：

- 在图 4-5 所示的智能菜单上勾选“启用编辑”和“启用删除”复选框。
- 将 AutoGenerateEditButton 属性设置为 True 以启用更新，将 AutoGenerateDeleteButton 属性设置为 True 以启用删除。
- 添加一个 CommandField，并将其 ShowEditButton 属性设置为 True 以启用更新，将其 ShowDeleteButton 属性设置为 True 以启用删除。

- 创建一个 TemplateField，其中 ItemTemplate 包含多个命令按钮，要进行更新时可将 CommandName 设置为“Edit”，要进行删除时可设置为“Delete”。有关更多信息，请参见“在 GridView Web 服务器控件中创建自定义列”。

GridView 控件可以显示一个用户界面(UI)，让用户能够编辑各行的内容。通常，可编辑的网格中会有一列包含一个按钮或链接，用户可以通过单击该按钮或链接将所在的行置于编辑模式下。(默认情况下，按钮标题是“编辑”。)

用户保存更改时，GridView 控件将更改和主键信息传递到由 DataSourceID 属性标识的数据源控件，从而调用适当的更新操作。例如，SqlDataSource 控件使用更改后的数据作为参数值来执行 SQL Update 语句。ObjectDataSource 控件调用其更新方法，并将更改作为参数传递给方法调用。

GridView 控件在 3 个字典集合中将值传递到数据源以进行更新或删除操作：Keys 字典、NewValues 字典和 OldValues 字典。可以使用传递到 GridView 控件的更新或删除事件的参数访问每个字典。

Keys 字典包含字段的名称和值，通过它们唯一标识将要更新或删除的记录，并始终包含键字段的原始值。若要指定哪些字段放置在 Keys 字典中，可将 DataKeyNames 属性设置为用逗号分隔的、用于表示数据主键的字段名称的列表。DataKeys 集合会用与为 DataKeyNames 属性指定的字段关联的值自动填充。DataKeyNames 属性中指定的字段的原始主键值存储在视图状态中。如果主键值中包含敏感信息，则应通过将页的 ViewStateEncryptionMode 属性设置为 Always 来加密视图状态的内容。

NewValues 字典包含正在编辑的行中的输入控件的当前值。OldValues 字典包含除键字段以外的任何字段的原始值，键字段包含在 Keys 字典中。

数据源控件使用 Keys、NewValues 和 OldValues 字典中的值作为更新或删除命令的参数。有关如何根据为绑定值创建的字典来创建数据源控件参数的信息，请参见“数据源控件如何为数据绑定字段创建参数”。

在通过处理 RowUpdating 或 RowDeleting 事件将任何这些字典的内容传递到数据源之前，可以对其进行检查或自定义。完成更新或删除后，GridView 控件会引发其 RowUpdated 或 RowDeleted 事件。这些事件允许执行查询后逻辑(如完整性检查)。

在完成更新或删除并引发所有事件之后，GridView 控件将重新绑定到数据源控件以显示已更新的数据。

GridView 控件中的可更新字段的原始值存储在 ViewState 中。如果在包含可更新 GridView 控件的 ASP .NET 页上禁用 ViewState，则开放式并发检查无法使用在 GridView 控件第一次绑定到数据源时检索到的可更新和主键字段的原始值。当该页为了执行更新或删除而进行回发时，数据库中的当前值将作为 GridView 控件中的可更新和主键字段的原始值被检索，因为 ViewState 中没有存储任何值。然后使用这些原始值执行更新或删除操作。如果原始值自从第一次填充 GridView 控件以来已更改，则更新或删除将会成功，但是开发式并发检查不会按预期的那样报告错误。

我们可以自定义编辑 (UI) 元素，如在每个数据字段的编辑模式下显示的控件类型。自动双向数据绑定允许自定义控件为数据存储区提供可编辑和已编辑的数据，以及从数据存储区获取可编辑和已编辑的数据。

如果更改数据源控件中的更新语句或重新排列 GridView 控件中的列，必须始终确保 GridView 控件传递到数据源的值与相应的数据源配置匹配。

表 5-1 列出了 GridView 控件中与数据编辑有关的事件。

表 5-1　GridView 控件中与数据编辑有关的事件

事 件 名	说　　明
RowDeleting	在单击 GridView 控件内某一行的“Delete”按钮(其 CommandName 属性设置为“Delete”的按钮)时发生，但在 GridView 控件从数据源删除记录之前。此事件通常用于取消删除操作
RowDeleted	在单击 GridView 控件内某一行的“Delete”按钮时发生，但在 GridView 控件从数据源删除记录之后。此事件通常用于检查删除操作的结果
RowEditing	在单击 GridView 控件内某一行的“Edit”按钮(其 CommandName 属性设置为“Edit”的按钮)时发生，但在 GridView 控件进入编辑模式之前。此事件通常用于取消编辑操作
RowCancelingEdit	在单击 GridView 控件内某一行的“Cancel”按钮(其 CommandName 属性设置为“Cancel”的按钮)时发生，但在 GridView 控件退出编辑模式之前。此事件通常用于停止取消操作
RowUpdating	在单击 GridView 控件内某一行的 Update 按钮(其 CommandName 属性设置为“Update”的按钮)时发生，但在 GridView 控件更新记录之前。此事件通常用于取消更新操作
RowUpdated	在单击 GridView 控件内某一行的 Update 按钮时发生，但在 GridView 控件更新记录之后。此事件通常用来检查更新操作的结果

ASP.NET GridView 控件有一个内置的选定内容功能，允许用户在网格中选择一行。在 GridView 控件中选择一行实际上不执行任何任务。但是，通过添加选定内容功能，可以向网格添加一些功能，在用户指向特定行时进行某种操作。将选定功能添加到 GridView 控件的典型用途包括以下两项：

- 用户选择某一行时，该行会以不同的外观重新显示。
- 用户选择某一行时，相关数据会在页中的其他位置显示，例如在 DetailsView 控件中显示。

在智能标记面板中选择“启用选定内容”，或设置 AutoGenerateSelectButton 属性为 True 都可以启用默认选定内容。

2. DetailsView 控件的基本概念与设置

DetailsView 控件是 ASP.NET 2.0 中另一个常用的数据处理控件，它的功能和 GridView 控件功能非常类似，同样具有编辑、删除、分页等功能，区别在于 DetailsView 控件每次仅显示一条记录，而 GridView 每次则可以显示多条记录，DetailsView 控件更适合于向数据库插入数据。

(1) DetailsView 简介

使用 DetailsView 控件，用户可以从它的关联数据源中一次显示、编辑、插入或删除一条记录。即使 DetailsView 控件的数据源公开了多条记录，该控件一次也仅显示一条数据

记录。默认情况下，DetailsView 控件将记录的每个字段显示在它自己的一行内。DetailsView 控件不支持排序。

DetailsView 控件可以自动对其关联数据源中的数据进行分页，若要启用分页，需将 AllowPaging 属性设置为 True。从关联的数据源选择特定的记录时，可以通过分页到该记录进行选择。由 DetailsView 控件显示的记录是当前选择的记录。

如图 5-1 所示，DetailsView 控件具有新建、编辑、删除、分页等功能。



图 5-1　DetailsView 控件操作界面

典型的 DetailsView 控件具有如下的类似代码：

```
<asp:DetailsView ID = "DetailsView1" runat = "server"
                AutoGenerateRows = "False" BackColor = "LightGoldenrodYellow"
                BorderColor = "Tan" BorderWidth = "1px" CellPadding = "2"
                 DataKeyNames = "title_id" DataSourceID = "SourceBookDetail"
                 ForeColor = "Black" GridLines = "None" Height = "50px" Width = "217px">
            <FooterStyle BackColor = "Tan" />
            <EditRowStyle BackColor = "DarkSlateBlue" ForeColor = "GhostWhite" />
            <PagerStyle BackColor = "PaleGoldenrod" ForeColor = "DarkSlateBlue"
                    HorizontalAlign = "Center" />
            <Fields>
             <asp:BoundField DataField = "title_id" HeaderText = "title_id"
                          ReadOnly = "True" SortExpression = "title_id" />
               <asp:BoundField DataField = "title" HeaderText = "title" SortExpression = "title" />
               <asp:BoundField DataField = "type" HeaderText = "type"
                            SortExpression = "type" />
               <asp:BoundField DataField = "pub_id" HeaderText = "pub_id"
                            SortExpression = "pub_id" />
               <asp:BoundField DataField = "price" HeaderText = "price"
                            SortExpression = "price" DataFormatString = "{0:c}" />
               <asp:BoundField DataField = "advance" HeaderText = "advance"
                            SortExpression = "advance" />
               <asp:BoundField DataField = "ytd_sales" HeaderText = "ytd_sales"
                            SortExpression = "ytd_sales" />
               <asp:BoundField DataField = "notes" HeaderText = "notes"
                            SortExpression = "notes" />
               <asp:BoundField DataField = "pubdate" HeaderText = "pubdate"
                            SortExpression = "pubdate" />
               <asp:TemplateField HeaderText = "Image">
                   <ItemTemplate>
                      <asp:Image ID = "Image1" runat = "server"
                 ImageUrl = '<% # Eval("image", "~\\images\\bookimages\\{0}") %>' />
                    </ItemTemplate>
               </asp:TemplateField>
            </Fields>
```

```
            <HeaderStyle BackColor = "Tan" Font - Bold = "True" />
            <HeaderTemplate>
                        Detail information of book
             </HeaderTemplate>
             <AlternatingRowStyle BackColor = "PaleGoldenrod" />
    </asp:DetailsView>
```

以上语法结构与 GridView 非常类似，其中不同处是字段的语法，在 GridView 中是 Columns，但是 DetailsView 则为 Fields。

（2）DetailsView 控件基本设置

DetailsView 属性大多与 GridView 类似，下面仅介绍不同的属性。

- DefaultMode：可以设置 DetailsView 的默认模式，用户执行 DetailsView 时的默认模式包括 Read Only（只读模式）、Edit（编辑模式）、Insert（新建模式）。
- Fields：DetailsView 的字段，相当于 GridView 的 Columns。

DetailsView 控件的字段称为 Field，包含在＜Fields＞＜/Fields＞标签中，表 5-2 列出了主要的字段类型。

表 5-2　DetailsView 字段说明

DetailsView 字段	说　　明
BoundField（数据绑定字段）	将数据以文字方式显示，默认字段
ButtonField（按钮字段）	可显示为 PushButton 或 LinkButton，单击产生 RowCommand 事件
CheckBoxField（CheckBox 字段）	将数据以 CheckBox 控件方式显示，数据类型为 Boolean 时适用
CommandField（命令字段）	显示编辑、删除、修改、选择时的按钮
HyperLinkField（超级链接字段）	将数据以 HyperLink 超链接方式显示
ImageField（图片字段）	将数据以图片方式显示
TemplateField（模板字段）	模板字段，可将字段替换为任何控件，并实现数据绑定

DetailsView 控件的模板字段有下列类型，如表 5-3 所示。

DetailsView 控件的字段模板比 GridView 多了新建模板（InsertTemplate），但是少了表尾模板（FooterTemplate）。DetailsView 控件还有 4 个表格模板，如表 5-4 所示。

表 5-3　DetailsView 模板字段

字段模板	说　明
ItemTemplate	项目模板
AlternatingItemTemplate	交错行样式
EditItemTemplate	编辑项目模板
HeaderTemplate	表头模板
InsertTemplate	新建模板

表 5-4　DetailsView 的表格模板

表格模板	说　明
EmptyDataTemplate	如果没有任何数据时显示的模板
PagerTemplate	分页按钮的模板
HeaderTemplate	表头模板
FooterTemplate	表尾模板

（3）在 DetailsView 控件中进行分页

DetailsView 控件具有一些内置的独特功能，允许用户一次一条地对记录分页，还支持自定义分页用户界面（UI）。在 DetailsView 控件中，一个数据页就是一个绑定记录。

如果 DetailsView 控件被绑定到某个数据源控件，则此控件将从数据源获取所有记录，

显示当前页的记录，并丢弃其余的记录。当用户移到另一页时，DetailsView 控件会重复此过程，显示另一条记录。

如果用户正使用 SqlDataSource 控件，并将其 DataSourceMode 属性设置为 DataReader，则 DetailsView 控件无法实现分页。

DetailsView 控件支持对其数据源中的记录进行分页，若要启用分页，只需将 AllowPaging 属性设置为 True，则在控件的最下端会添加页码显示，如图 5-22 所示。

用户可以用多种方式自定义 DetailsView 分页的用户界面。在将 AllowPaging 属性设置为 True 时，PagerSettings 属性允许自定义 DetailsView 控件生成的分页用户界面的外观。DetailsView 控件可显示允许向前和向后导航的方向控件以及允许用户移动到特定页的数字控件。

DetailsView 控件的 PagerSettings 属性设置为一个 PagerSettings 类。可以通过将 DetailsView 控件的 Mode 属性设置为 PagerButtons 值来自定义分页模式。例如，用户可以通过以下代码来自定义分页用户界面模式：

```
DetailsView1.PagerSettings.Mode = PagerButtons.NextPreviousFirstLast;
```

也可以通过属性窗口设置 Mode 属性来指定分页用户界面，可用的模式如表 5-5 所示。

表 5-5 PagerSettings. Mode 属性

Mode 设置	说　明	图　例
NextPrevious	只有上一页、下一页	<>
Numeric	只显示数字	123456
NextPreviousFirstLast	显示第一页、上一页、下一页、最后一页	<<<>>>
NumericFirstLast	显示上一页、下一页、数字	<<…456…>>

（4）使用 DetailsView 控件修改数据

DetailsView 控件具有一项内置功能，可让用户能够无须编程即可编辑、删除和插入记录，用户可以使用事件和模板来自定义 DetailsView 控件的编辑功能。在 DetailsView 控件中显示来自数据源的单条记录的值，其中每个数据行表示该记录的一个字段。

用户可以通过将 AutoGenerateEditButton、AutoGenerateInsertButton 和 AutoGenerateDeleteButton 属性中的一个或多个设置为 True，来启用 DetailsView 控件的内置编辑功能，如图 5-2 所示。DetailsView 控件将自动添加此功能，使用户能够编辑或删除当前绑定的记录以及插入新记录，但前提是 DetailsView 控件的数据源支持编辑。

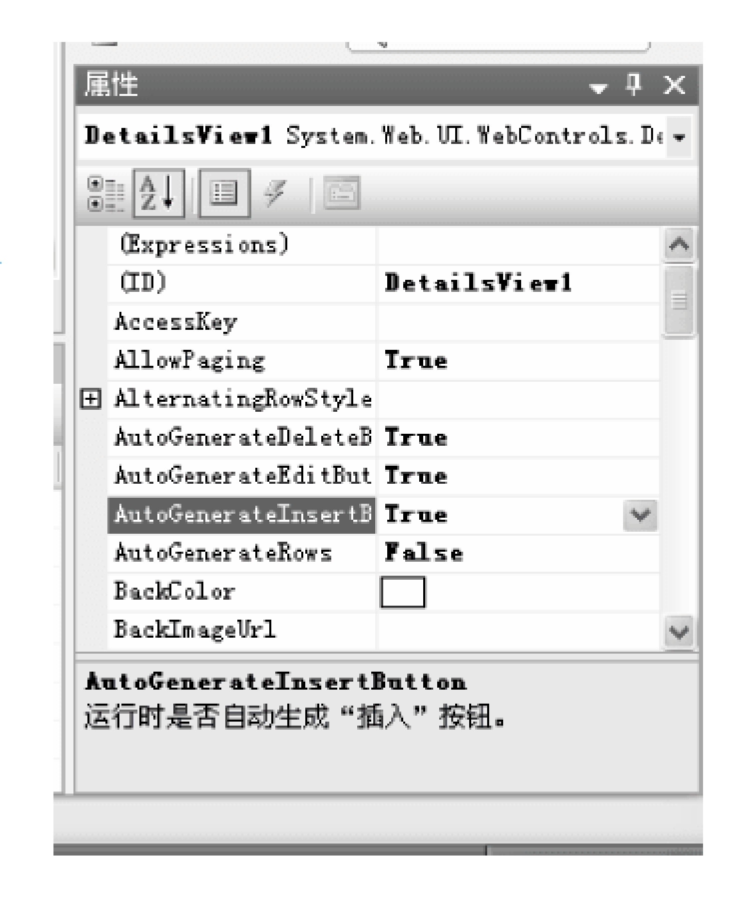


图 5-2 启用 DetailsView 内置编辑功能

DetailsView 控件提供了一个用户界面，使用户能够修改绑定记录的内容。通常，在一个可编辑视图中会显示一个附加行，其中包含“编辑”、“新建”和“删除”命令按钮。默认情况下，这一行会添加到 DetailsView 控件的底部。

当用户单击某个命令按钮时，DetailsView控件会重新显示该行，并在其中显示可让用户修改该行内容的控件，编辑按钮将被替换为可让用户保存更改或取消编辑行的按钮。在完成更新后，DetailsView控件会引发ItemUpdated事件。此事件使我们有机会执行数据库更新后的必要代码，如完整性检查。同样，DetailsView控件会在完成插入后引发其ItemInserted事件，而在完成删除后引发其ItemDeleted事件。在完成更新且已引发所有事件后，DetailsView控件会重新绑定到数据源控件以显示更新后的数据。

(5) DetailsView控件常用事件

DetailsView控件可引发一些事件，这些事件在当前记录显示或更改时发生。当单击一个命令控件(如作为DetailsView控件的一部分的Button控件)时也会引发事件，表5-6列出了DetailsView控件的常用事件。

表5-6　DetailsView控件常用事件

事件名称	说　明
ItemCommand	在单击DetailsView控件中的某个按钮时发生
ItemCreated	在DetailsView控件中创建记录时发生
ItemDeleting	在单击DetailsView控件中的Delete按钮时发生，但在删除操作之前
ItemDeleted	在单击DetailsView控件中的Delete按钮时发生，但在删除操作之后
ItemInserting	在单击DetailsView控件中的Insert按钮时发生，但在插入操作之前
ItemInserted	在单击DetailsView控件中的Insert按钮时发生，但在插入操作之后
ItemUpdating	在单击DetailsView控件中的Update按钮时发生，但在更新操作之前
ItemUpdated	在单击DetailsView控件中的Update按钮时发生，但在更新操作之后
ModeChanging	在DetailsView控件试图在编辑、插入和只读模式之间切换时发生，但在更新CurrentMode属性之前
ModeChanged	在DetailsView控件试图在编辑、插入和只读模式之间切换时发生，但在更新CurrentMode属性之后

5.4 项目实施

5.4.1 任务5-1 使用GridView控件实现图书信息修改功能

1. 任务介绍

在本任务中，实现对图书信息的修改、删除功能。

2. 任务分析

GridView控件具有一些内置功能，允许在不需要编程的情况下编辑或删除记录，下面就用它来实现对图书信息的修改。

(1) 创建显示图书信息的Web页面

首先，我们来创建一个存放有关产品信息Web页的文件夹manage，这也对应着系统设计中的一个功能模块。

采用和4.4.1小节任务4-1中步骤1相同的方法，创建一个Web页面BooksInfoEdit.aspx，

在页面中拖放入一个 GridView 控件和一个 SqlDataSource 控件，如图 5-3 所示。

Content - Content1 (自定义)

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

SqlDataSource - SqlDataSource1

图 5-3　BooksInfoEdit. aspx 页面

(2) 设置 SqlDataSource 数据源

单击 SqlDataSource 控件的智能按钮，在弹出的菜单中选择“配置数据源…”，打开“配置数据源”对话框，在可选数据源下拉列表中选中已建立的连接 NetBookConnectionString，单击“下一步”按钮，进到“配置数据源”对话框的“配置 Select 语句”步骤，选中 t_book 表，勾选除“ * ”外所有字段，如图 5-4 所示。



图 5-4　“配置 Select 语句”对话框

单击“高级…”按钮，弹出“高级 SQL 生成选项”对话框，如图 5-5 所示。

选中“生成 INSERT、UPDATE 和 DELETE 语句”复选框，单击“确定”按钮。然后依次单击“下一步”按钮，直到完成配置。

查看页面的源试图，系统生成的 SqlDataSource 控件的源代码如下所示。

```
< asp: SqlDataSource ID = " SqlDataSource1 " runat = " server " ConnectionString = " <% $
ConnectionStrings:NetBookConnectionString %>"
    SelectCommand = "SELECT [bookid], [bookname], [ISBN], [author], [publisher], [pubdate],
[pubnumber], [price], [NetBookPrice], [picture], [memo], [amount], [secondCategoryId],
[recommendFlag], [recommendDate], [recommendMemo], [NetBookDate], [CategoryId],
```

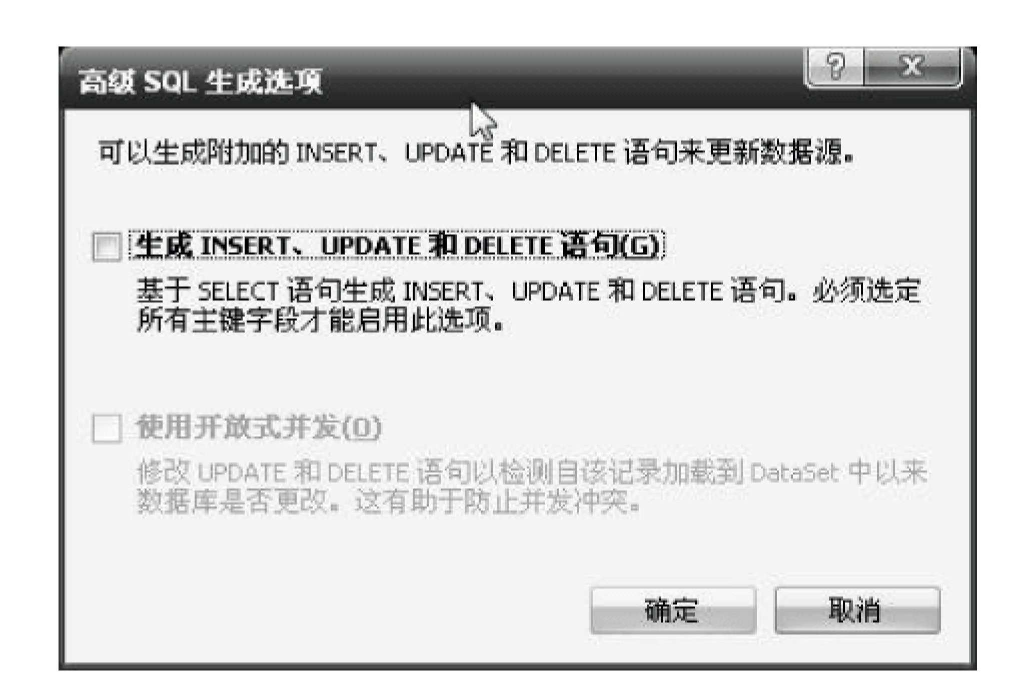


图 5-5　“高级 SQL 生成选项”对话框

```
[discount], [saleamount] FROM [t_book]"
    DeleteCommand = "DELETE FROM [t_book] WHERE [bookid] = @bookid"
    InsertCommand = "INSERT INTO [t_book] ([bookname], [ISBN], [author], [publisher],
[pubdate], [pubnumber], [price], [NetBookPrice], [picture], [memo], [amount],
[secondCategoryId], [recommendFlag], [recommendDate], [recommendMemo], [NetBookDate],
[CategoryId], [discount], [saleamount]) VALUES (@bookname, @ISBN, @author, @publisher,
@pubdate, @pubnumber, @price, @NetBookPrice, @picture, @memo, @amount,
@secondCategoryId, @recommendFlag, @recommendDate, @recommendMemo, @NetBookDate,
@CategoryId, @discount, @saleamount)"
    UpdateCommand = "UPDATE [t_book] SET [bookname] = @bookname, [ISBN] = @ISBN,
[author] = @author, [publisher] = @publisher, [pubdate] = @pubdate, [pubnumber] =
@pubnumber, [price] = @price, [NetBookPrice] = @NetBookPrice, [picture] = @picture,
[memo] = @memo, [amount] = @amount, [secondCategoryId] = @secondCategoryId,
[recommendFlag] = @recommendFlag, [recommendDate] = @recommendDate, [recommendMemo] =
@recommendMemo, [NetBookDate] = @NetBookDate, [CategoryId] = @CategoryId, [discount] =
@discount, [saleamount] = @saleamount WHERE [bookid] = @bookid">
            <DeleteParameters>
            <asp:Parameter Name="bookid" Type="Int32" />
            </DeleteParameters>
            <UpdateParameters>
            <asp:Parameter Name="bookname" Type="String" />
            <asp:Parameter Name="ISBN" Type="String" />
            <asp:Parameter Name="author" Type="String" />
            <asp:Parameter Name="publisher" Type="String" />
            <asp:Parameter DbType="Date" Name="pubdate" />
            <asp:Parameter Name="pubnumber" Type="Int32" />
            <asp:Parameter Name="price" Type="Decimal" />
            <asp:Parameter Name="NetBookPrice" Type="Decimal" />
            <asp:Parameter Name="picture" Type="String" />
            <asp:Parameter Name="memo" Type="String" />
            <asp:Parameter Name="amount" Type="Int32" />
            <asp:Parameter Name="secondCategoryId" Type="Int32" />
            <asp:Parameter Name="recommendFlag" Type="Boolean" />
            <asp:Parameter DbType="Date" Name="recommendDate" />
            <asp:Parameter Name="recommendMemo" Type="String" />
            <asp:Parameter DbType="Date" Name="NetBookDate" />
            <asp:Parameter Name="CategoryId" Type="Int32" />
```

```
            <asp:Parameter Name = "discount" Type = "Decimal" />
            <asp:Parameter Name = "saleamount" Type = "Int32" />
            <asp:Parameter Name = "bookid" Type = "Int32" />
        </UpdateParameters>
        <InsertParameters>
            <asp:Parameter Name = "bookname" Type = "String" />
            <asp:Parameter Name = "ISBN" Type = "String" />
            <asp:Parameter Name = "author" Type = "String" />
            <asp:Parameter Name = "publisher" Type = "String" />
            <asp:Parameter DbType = "Date" Name = "pubdate" />
            <asp:Parameter Name = "pubnumber" Type = "Int32" />
            <asp:Parameter Name = "price" Type = "Decimal" />
            <asp:Parameter Name = "NetBookPrice" Type = "Decimal" />
            <asp:Parameter Name = "picture" Type = "String" />
            <asp:Parameter Name = "memo" Type = "String" />
            <asp:Parameter Name = "amount" Type = "Int32" />
            <asp:Parameter Name = "secondCategoryId" Type = "Int32" />
            <asp:Parameter Name = "recommendFlag" Type = "Boolean" />
            <asp:Parameter DbType = "Date" Name = "recommendDate" />
            <asp:Parameter Name = "recommendMemo" Type = "String" />
            <asp:Parameter DbType = "Date" Name = "NetBookDate" />
            <asp:Parameter Name = "CategoryId" Type = "Int32" />
            <asp:Parameter Name = "discount" Type = "Decimal" />
            <asp:Parameter Name = "saleamount" Type = "Int32" />
        </InsertParameters>
</asp:SqlDataSource>
```

从源代码可以看出，设置工具为自动生成了 SelectCommand、DeleteCommand、InsertCommand 和 UpdateCommand 的对应 SQL 语句，并且为每个语句创建了必要的参数。因为在本任务中不涉及插入命令，所以可以把所有有关 Insert 的内容（灰底纹部分）删去。由于 discount 字段是计算字段，不可编辑，故也必须在 UpdateCommand 命令中将其删除（即将带有下划线部分删除）。

（3）设置 GridView 控件

配置好 SqlDataSource 数据源控件后，就要设置 GridView 控件，作为用户操作数据库，修改图书信息的界面。

单击 GridView 控件的智能按钮，弹出“GridView 任务”菜单，在自动套用格式对话框选择“彩色”选项，在“选择数据源”下拉列表中选择先前配置好的“SqlDataSource1”。

由于选择了数据源，系统根据所选数据源控件的配置情况，增添了可选功能，更新了“GridView 任务”菜单，下面要设置 GridView 控件的显示外观，先单击“编辑列…”按钮，弹出“字段”对话框，如图 5-6 所示。

在图 5-6 中，一方面对所有绑定字段进行属性设置，调整它们的外观和格式，另外还要添加命令字段。单击“可用字段”下拉列表中 CommandField 左边展开符，选中“编辑、更新、取消”选项，单击“添加”按钮。再选中“删除”选项，单击“添加”按钮。如果有些绑定字段不需要在此编辑，可以把它们从“选定的字段”列表中删除。

勾选智能菜单中的“启用排序”和“启用分页”复选框。经过编辑，GridView 控件如图 5-7 所示。

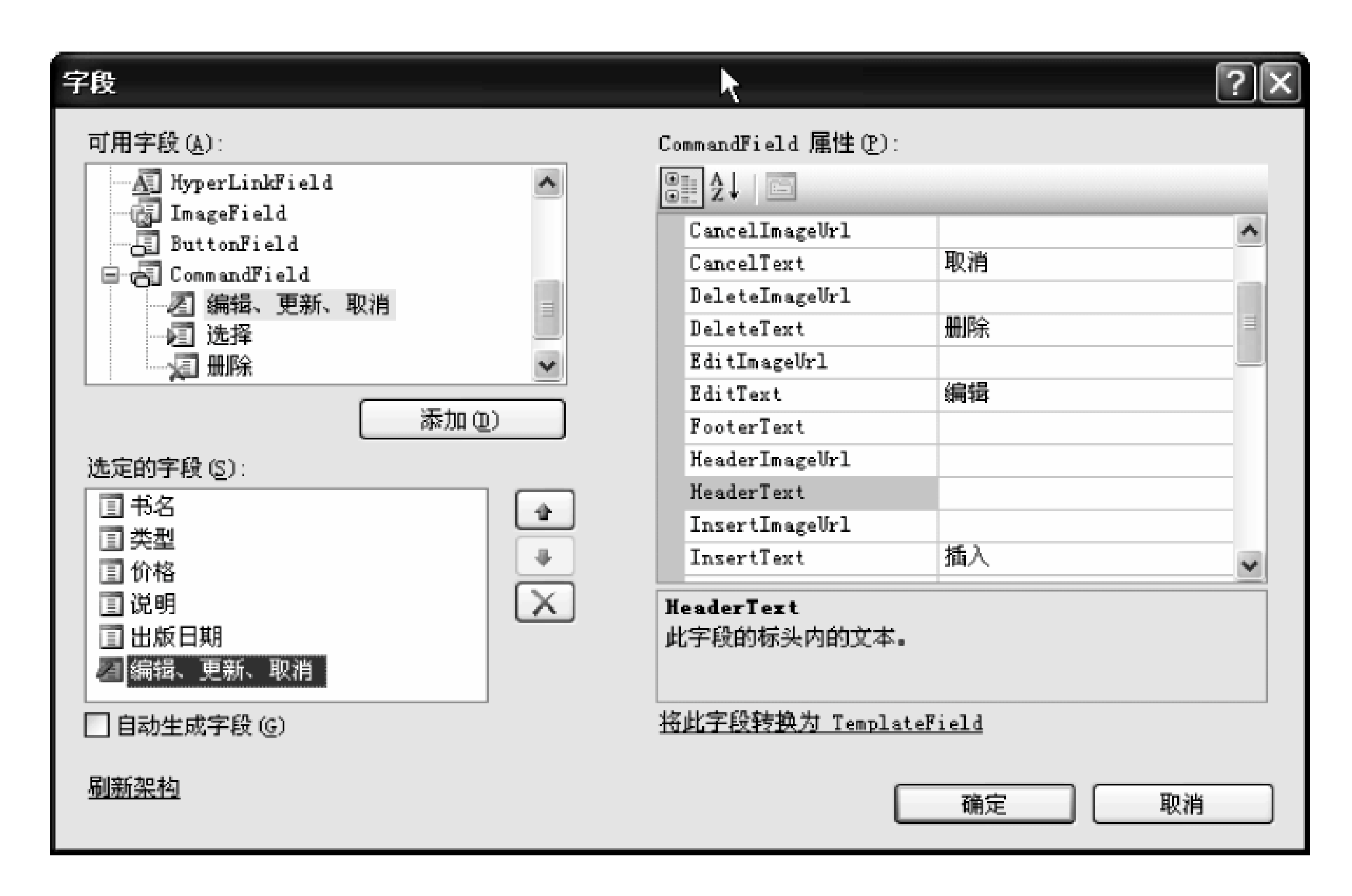


图 5-6　“字段”对话框

编号	书名	作者	出版社	出版日期	价格	封面	简介	是否推荐	则扣率		
0	abc	abc	abc	2009年8月12日	￥0.00	abc	abc	☐	0.00	编辑	删除
1	abc	abc	abc	2009年8月12日	￥0.10	abc	abc	☑	0.10	编辑	删除
2	abc	abc	abc	2009年8月12日	￥0.20	abc	abc	☐	0.20	编辑	删除
3	abc	abc	abc	2009年8月12日	￥0.30	abc	abc	☑	0.30	编辑	删除
4	abc	abc	abc	2009年8月12日	￥0.40	abc	abc	☐	0.40	编辑	删除
5	abc	abc	abc	2009年8月12日	￥0.50	abc	abc	☑	0.50	编辑	删除
6	abc	abc	abc	2009年8月12日	￥0.60	abc	abc	☐	0.60	编辑	删除
7	abc	abc	abc	2009年8月12日	￥0.70	abc	abc	☑	0.70	编辑	删除
8	abc	abc	abc	2009年8月12日	￥0.80	abc	abc	☐	0.80	编辑	删除
9	abc	abc	abc	2009年8月12日	￥0.90	abc	abc	☑	0.90	编辑	删除
1 2											

图 5-7　经过编辑的 GridView 控件

按 F5 键调试程序，将显示如图 5-8 所示的运行时效果。



编号	书名	作者	出版社	出版日期	价格	封面	简介	是否推荐	则扣率		
1	明朝那些事儿	当年明月	中国海关	2002年3月1日	￥23.00	image/1.jpg	明朝历史上的最后一位皇帝，自来有许多传奇。关于崇祯帝究竟是一个昏庸无能的皇帝，还是一个力...	☐	7.83	编辑	删除
2	牧羊少年奇幻之旅	（巴西）柯艾略	南海	2003年4月1日	￥25.00	image/2.jpg	迄今唯一一部语种超过《圣经》的书　在巴西，按知名度他与上帝、足球并列　在美国，他是唯...	☐	8.00	编辑	删除
3	村级干部	贺享雍	上海	2002年4月8日	￥16.00	image/3.jpg	一座百年老屋，见证几代恩仇；两个坚强女性，撬动僻静山村。跌宕起伏的故事情节，变化多端的人物命...	☐	8.75	编辑	删除

图 5-8　运行时初始效果

在运行时的初始状态，GridView 控件显示了图书信息，每条记录一行，与以往不同的是，在每条记录的右侧，有两个链接，分别是“编辑”和“删除”。它们就是命令字段 CommandField 中的两个，因为它们的属性 ButtonType 默认是“Link”，所以显示为超链接，我们也可以把它们改为按钮(Button)或图像(Image)。

当我们要编辑某条记录时，只需单击“编辑”链接，就可以使该条记录进入编辑状态，如图 5-9 所示。

编号	书名	作者	出版社	出版日期	价格	封面	简介	是否推荐	到扣率		
1	明朝那些事儿	当年明月	中国海关	2002年2月1日	¥23.00	image/1.jpg	明朝历史上的最后一位皇帝	☐	7.83	更新	取消
2	牧羊少年奇幻之旅	(巴西)柯艾略	南海	2003年4月1日	¥25.00	image/2.jpg	迄今唯一一部销量超过《圣经》的书 在巴西，被知名度他与上帝、足球并列 在美国，他是唯...	☐	8.00	编辑	删除
3	村级干部	贺享雍	上海	2002年4月8日	¥16.00	image/3.jpg	一座百年老屋，见证几代恩仇；两个坚强女性，撼动僻静山村。跌宕起伏的故事情节，变化多端的人物命...	☐	8.75	编辑	删除
4	贫民窟的百万富翁	(印)斯瓦鲁普	巴西	2002年2月12日	¥34.00	image/4.jpg	十八岁的酒吧服务员罗摩，生活在孟买的贫民窟里。他参加了一个名为《谁将赢得十个亿》的电视知识问...	☐	8.82	编辑	删除

图 5-9　GridView 控件的编辑状态

处于编辑状态的记录，每个字段如果是布尔值，就有一个复选框用来输入编辑值，如果是非布尔值，则有一个文本输入框用来输入编辑值。记录的右侧有两个链接，分别是“更新”和“取消”。当编辑完成后，如果单击“更新”链接，就用新编辑的值更新数据库，如果单击“取消”链接，就放弃本次编辑。

如果我们单击某条记录右边的超链接“删除”，就会删除这条记录。

不管是编辑还是删除，对数据库的操作都有成功和失败两种可能。为了向用户准确地告知操作的结果，我们可以利用相应的 GridView 事件，来完善我们的操作。GridView 控件关于数据操作完整的事件列表，见表 5-1。

下面以删除数据为例，介绍事件响应代码。假设在页面中插入一个标签控件，ID 为 Message，作用是显示操作结果。显示图书信息的 GridView 控件的 ID 是 BookInfoGridView，它的响应删除操作的代码如下：

```
void BookInfoGridView_RowDeleted(Object sender, GridViewDeletedEventArgse) {
    // 操作成功
    if(e.Exception== null) {
        Message.Text = "记录被成功删除";
    }
    //操作失败
    else {
        Message.Text = "有错误产生,记录违背删除";
        e.ExceptionHandled = true;
    }
}
```

3. 任务完成总结

本任务要实现网站图书管理的功能，即修改、删除相关的图书信息。利用控件具有的内置功能，可以非常方便地编辑或删除记录，判断操作是否成功，可以用 GridView 控件的相

应事件来进行检测。

4. 课堂训练与知识拓展

GridView 控件处于编辑状态时，默认情况下各字段用一个复选框或文本输入框来输入编辑值。在有些情况下，用户要输入的值是某个列表中的一个，例如输入省份、职业等，这时最好的方案是采用下拉列表框。如果某个字段要采用非默认控件输入，可以将这个字段改为模板字段，在模板中加入所需的控件，并将选定值绑定到对应字段。

5.4.2 任务 5-2 使用 DetailsView 控件实现图书信息浏览、修改、删除和增加功能

1. 任务介绍

在实际应用中，图书的数目往往非常庞大，而且每条记录的具体字段又很多，采用任务 5-1 的方法往往不能完全满足需求，因此，在本任务中就要采用另外更加有效的方法实现对图书信息的修改、删除和增加功能。

2. 任务分析

在本任务中要采用更加有效的方法实现对图书信息的修改、删除和增加，使用 DetailsView 控件是最好的选择。

采用 DetailsView 控件，可以从它的关联数据源中一次显示、编辑、插入或删除一条记录。默认情况下，DetailsView 控件将记录的每个字段显示在它自己的一行内。DetailsView 控件通常用于更新和插入新记录，并且通常在主/从方案中使用，在这些方案中，主控件的选中记录决定要在 DetailsView 控件中显示的记录。即使 DetailsView 控件的数据源返回了多条记录，该控件一次也仅显示一条数据记录。

DetailsView 控件依赖于数据源控件的功能执行诸如更新、插入和删除记录等任务。

(1) 创建包含 DetailsView 控件的 Web 页面

首先，在文件夹 manage 下创建一个 Web 页面 BookDetailsEdit.aspx，在页面中拖放入一个 GridView 控件、一个 DetailsView 控件和两个 SqlDataSource 控件，如图 5-10 所示。两个 SqlDataSource 控件分别命名为 BookSource 和 DetailsSource。

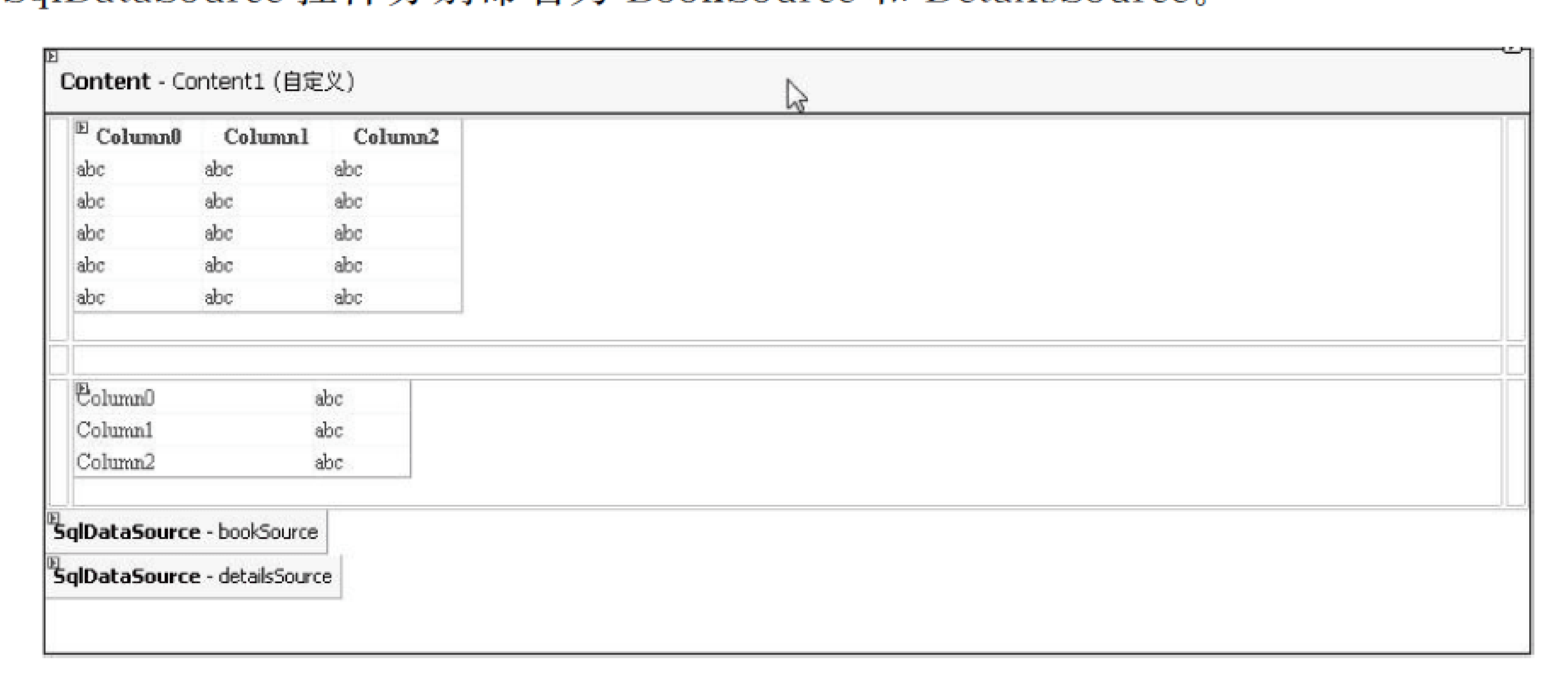


图 5-10 BookDetailsEdit.aspx 页面

(2) 设置显示图书概要信息的数据源

首先单击 bookSource 控件的智能按钮,在弹出的菜单中选择“配置数据源…”,打开“配置数据源”对话框,在可选数据源下拉列表中选中先前建立的连接“NetBookConnectionString”,单击“下一步”按钮,进到“配置数据源”对话框的“配置 Select 语句”步骤,选中 t_book 表,勾选几个重要字段,如图 5-11 所示。



图 5-11 bookSource 控件的“配置 Select 语句”步骤

依次单击“下一步”按钮,直到完成配置。

(3) 设置显示图书概要信息的控件

配置好显示图书概要信息的数据源控件 bookSource 后,就要设置显示图书概要信息的 GridView 控件,让用户浏览信息,选中要编辑的记录。

单击 GridView 控件的智能按钮,弹出“GridView 任务”菜单,在自动套用格式对话框中选择“彩色”选项,在“选择数据源”下拉列表中选择先前配置好的“bookSource”。

由于选择了数据源,系统根据所选数据源控件的配置情况,增添了可选功能,更新了“GridView 任务”菜单。下面设置 GridView 控件的显示外观,先单击“编辑列…”按钮,弹出“字段”对话框,如图 5-12 所示。

在图 5-12 中,一方面对所有绑定字段进行属性设置,调整它们的外观和格式,另外还要添加命令字段。单击“可用字段”下拉列表中 CommandField 左边展开符,选中“选择”选项,单击“添加”按钮。

勾选智能菜单中的“启用排序”和“启用分页”复选框。经过编辑,GridView 控件如图 5-13 所示。

(4) 设置显示图书详细信息的数据源

采用和任务 5-1 中步骤 4 完全一样的方法,配置显示图书详细信息的数据源控件 detailsSource,这次在源代码中只需要删除 UpdateCommand 和 InsertCommand 命令中的 discount 部分。

因为要显示详细数据的只有一条记录,也就是用户在 GridView 控件中选定的记录,所



图 5-12 显示图书概要信息的 GridView 控件的“字段”对话框

图 5-13 显示图书概要信息的 GridView 控件

以必须在 SelectCommand 中增加一个限制条件。在配置 Selecct 语句的对话框中，单击“WHERE…”按钮，弹出“添加 WHERE 子句”对话框，如图 5-14 所示。

在“列”下拉列表中选择“bookid”字段；在“运算符”下拉列表中选择“=”；在“源”下拉列表中选择“Control”，表示 SQL 参数将由控件来提供；在“控件 ID”下拉列表中选择显示图书概要信息的 GridView1。单击“添加”按钮，系统在“WHERE 子句”文本框中生成相应的 SQL 表达式，如图 5-15 所示。

单击“确定”按钮，然后依次单击“下一步”按钮直到完成整个配置。



图 5-14 “添加 WHERE 子句”对话框

(5) 设置显示图书详细信息的显示控件

单击 DetailsView 控件的智能按钮,弹出如图 5-16 所示的菜单。



图 5-15 自动生成的 SQL 表达式



图 5-16 DetailsView 控件的智能菜单

在自动套用格式对话框中选中“彩色”选项,在“选择数据源”下拉列表选择先前配置好的“detailsSource”。由于选择了数据源,系统根据所选数据源控件的配置情况,增添可选功能,更新菜单。

下面设置 DetailsView 控件的显示外观。单击“编辑字段…”菜单项,弹出“字段”对话框,如图 5-17 所示。

在图 5-17 中,一方面对所有绑定字段进行属性设置,调整它们的外观和格式,另外还要添加命令字段。单击“可用字段”下拉列表中 CommandField 左边展开符,分别选中“编辑、更新、取消”选项、“删除”选项和“新建、插入、取消”选项,单击“添加”按钮。

经过编辑,DetailsView 控件如图 5-18 所示。

按 F5 键调试程序,将显示如图 5-19 所示的运行时效果。

因为还没选中任何图书,所以在初始界面上只是显示了 GridView 控件,列出了图书的概要信息,当选中其中任意一本书时,将显示如图 5-20 所示的结果。



图 5-17　DetailsView 控件的“字段”对话框



图 5-18　经过设置的 DetailsView 控件

书名	ISBN	作者	价格	
明朝那些事儿	9787801655998	当年明月	￥23.00	选择
牧羊少年奇幻之旅	9787801634583	（巴西）柯艾略	￥25.00	选择
村级干部	9787333633283	贺享雍	￥16.00	选择
贫民窟的百万富翁	2354566777776	（印）斯瓦鲁普 .	￥34.00	选择
致命的温柔	2356678866555	艾米	￥35.00	选择
1 2 3 4 5 6				

图 5-19　任务 5-2 运行的初始效果

书名	ISBN	作者	价格	
明朝那些事儿	9787801655998	当年明月	￥23.00	选择
牧羊少年奇幻之旅	**9787801634583**	**（巴西）柯艾略**	**￥25.00**	**选择**
村级干部	9787333633283	贺享雍	￥16.00	选择
贫民窟的百万富翁	2354566777776	（印）斯瓦鲁普 .	￥34.00	选择
致命的温柔	2356678866555	艾米	￥35.00	选择
1 2 3 4 5 6				

图书编号	2
书名	牧羊少年奇幻之旅
ISBN	9787801634583
作者	（巴西）柯艾略
出版商	南海
出版日期	2003年4月1日
版次	1002
价格	￥25.00
网上价格	￥20.00
封面	image/2.jpg
简介	迄今唯一一部语种超过《圣经》的书　在巴西，按知名度他与上帝、足球并列　在美国，他是唯...
总数	51
次级目录Id	2
是否推荐	
推荐日期	
推荐内容	
上网日期	
目录Id	1
折扣	8.00
销售总数	3
编辑	
删除	
新建	

图 5-20　任务 5-2 选中图书时运行的结果

我们可以看到，在 DetailsView 控件中显示出了选中图书的详细信息，按照默认格式，每个字段占一行。最后 3 行是按钮，分别执行“编辑”、“删除”和“新建”命令。

如果单击“编辑”按钮，将转到编辑状态，如图 5-21 所示。

图书编号	2
书名	牧羊少年奇幻之旅
ISBN	9787801634583
作者	（巴西）柯艾略
出版商	南海
出版日期	2003-4-1 0:00:00
版次	1002
价格	25.0000
网上价格	20.0000
封面	image/2.jpg
简介	迄今唯一一部语种超过《
总数	51
次级目录Id	2
是否推荐	☐
推荐日期	
推荐内容	
上网日期	
目录Id	1
折扣	8.00
销售总数	3
更新 取消	

图 5-21 DetailsView 控件的编辑状态

处于编辑状态的记录，每个字段如果是布尔值，就有一个复选框用来输入编辑值，如果是非布尔值，则有一个文本输入框用来输入编辑值。控件的最后一行有两个按钮，分别是“更新”和“取消”。当编辑完成后，如果单击“更新”按钮，就用新编辑的值更新数据库，如果单击“取消”按钮，就放弃本次编辑。不管单击哪个按钮，DetailsView 控件将转到如图 5-20 所示的显示界面。

如果单击显示界面的“删除”按钮，将会删除这条记录。

如果单击显示界面的“新建”按钮，将转到新建状态，如图 5-22 所示。

处于新建状态的记录，每个字段如果是布尔值，就有一个复选框用来输入编辑值，如果是非布尔值，则有一个文本输入框用来输入编辑值。控件的最后一行有两个按钮，分别是“插入”和“取消”。当内容填写完成后，如果单击“插入”按钮，就用新填写的值生成一条记录插入数据库，如果单击“取消”按钮，就放弃本次新增操作。

不管是编辑、删除还是新增，对数据库的操作都有成功和失败两种可能。为了准确地告知用户操作的结果，可以利用相应的 DetailsView 事件，来完善我们的操作。DetailsView 控件常用事件的列表，见表 5-6。

为了防止在编辑和新增记录时用户输入恶意代码，入侵或破坏网站，在更新及新增记录时可以利用相应事件进行预防。下面以更新数据为例，介绍事件响应代码。

```
void CustomerDetail_ItemUpdating(object sender, DetailsViewUpdateEventArgs e) {
    // 遍历用户输入的每一个值
    for (int i = 0; i < e.NewValues.Count; i++) {
      if (e.NewValues[i] != null) {
        e.NewValues[i] = Server.HtmlEncode(e.NewValues[i].ToString());
```

```
            }
        }
    }
```

ISBN	
书名	
作者	
出版日期	
出版商	
版次	
价格	
网上价格	
封面	
简介	
总数	
推荐日期	
是否推荐	
推荐内容	
上网日期	
次级目录Id	
目录Id	
折扣	
销售总数	
插入 取消	

图 5-22　DetailsView 控件的新建状态

3. 任务完成总结

在实际应用中，图书的数目往往非常庞大，而且每条记录的具体字段又很多，比较好的发布方法是先用 GridView 控件显示出当前图书的概要信息，由用户选定某条记录，再用 DetailsView 控件显示它的详细情况，用户还可以在 DetailsView 控件中编辑、插入或删除一条记录。

默认情况下，DetailsView 控件将记录的每个字段显示在它自己的一行内。DetailsView 控件通常用于更新和插入新记录。

在操作数据库的过程中，也可以用 DetailsView 控件的事件来进行操作检测和预防恶意输入。

4. 课堂训练与知识拓展

DetailsView 控件是个功能强大、使用方便的控件，它的默认模式是只读模式(ReadOnly)，也就是数据显示模式，一般以主/从页面的形式出现。但也可以设置成编辑(Edit)或插入(Insert)模式，在设置成插入(Insert)模式时，因不用显示某个记录，就可单独出现。另外，它还有丰富的属性、方法和事件，为我们创建复杂应用提供了方便。

5.5 项目总结

创建网上书店网站,随着销售等业务活动的不断进行,必须不断修改原先的图书信息,增添新的内容,要实现网站图书管理的功能。

图书管理的核心是访问数据库,查询、修改、插入、删除数据。

对数据库的连接和 SQL 语句设置在 SqlDataSource 控件中进行。

进行查询、修改、删除操作,可以使用 GridView 控件,也可以使用 DetailsView 控件,DetailsView 控件通常用于更新和插入新记录。

5.6 项目实训

1. 任务描述

当创建了网上书店网站后,在向用户发布了各种图书信息的同时,也需发布各种类别目录,以方便用户查询。随着业务的发展,还必须能不断修改原先的类别信息,增添新的内容,实现网站图书类别管理的功能。

2. 任务要求

① 创建页面供用户浏览信息。
② 创建页面供用户编辑类别内容。
③ 创建页面供用户新增类别记录。

项目6

实现网上书店读者注册管理功能

6.1 项目介绍

建立了网上书店网站后，总是要考虑哪些资源（狭义的认为就是页面）是可以直接访问的，哪些资源是需要限制访问的，例如，只能对那些留有通信信息的用户开放在线订购功能。这就是说，需要一种机制能够判断哪些用户才是一个合法的用户（注册过的用户，一般需要经过“登录认证”），哪些是普通用户（匿名用户），这种区分机制就是身份验证机制。在身份验证的基础上，还可以确定哪些特定用户可以使用哪些特定资源，即授权，这将在以后的章节中详细介绍。

从用户的角度来说，这种身份验证机制使得用户能够在网站上注册登记自己的相关信息，以获得特定服务。当这种用户信息被注册登记后，用户必须能够查阅和修改。

用户身份的验证在合理的时间段内只需进行一次，也就是说验证后的合法用户可以访问网站内所有的合法资源，而不是每访问一个资源就验证一次。

通过用户名和密码，对用户进行身份验证，并指派其可访问的资源，这部分工作一直都是网站开发的重要内容，也是本项目将重点介绍和实现的内容。

6.2 项目分析

在本项目中将实现的主要功能是用户注册管理功能，即用户注册和用户管理功能。

当一个普通用户在网站上选择注册功能时，将会出现一个信息输入界面，主要包含将要使用的用户名以及密码，这两项是区分不同注册用户的必不可少的信息。为了更好地为用户提供服务，注册时还可以提供 E-mail 地址、真实姓名等辅助信息。为了预防注册时密码设置出现误操作，密码输入要求进行两次，只有两次输入相同的字符串，才是合法的密码。有些已注册的用户会遗忘密码，这就需要一个恢复机制，使系统能不通过密码而识别注册用户。常用的解决方案是在注册时用户提供专门问题和答案，由于这些问题和答案具有实际意义，用户很容易记住。当用户忘记了密码时，如果能正确回答专门设置的问题，就认为是注册用户。

当一个用户注册后，下一次访问网站只需要告诉系统“我是谁”，即可登录，这需要输入自己的用户名和密码。系统可据此确认是否是注册的合法用户。对于不同性质的用户，系统还可以提供不同的界面和功能。例如，对于注册用户，可以提供含有当前用户名的欢迎词

以及退出登录功能；对于非注册用户，可以提供到注册页面的链接。

对于绝大多数商业网站，用户的注册和验证，其核心就是对数据库的访问。当一个用户注册时，也就是将一些经过组织的用户数据保存到指定的数据库。当用户登录时，就是根据用户名和密码在数据库中查找合法的记录，如果找到，则登录成功，否则就是登录失败。围绕着这些注册、登录过程，还有许多界面设计与组织、数据验证、数据加密等辅助工作。可以从底层开始开发自己的用户注册、登录模块，实现对所有数据库的访问、数据验证、加密等。在分布式程序开发的早期阶段，这些确实是开发人员的重要工作，但是对每个网站而言，这些工作都非常类似，结果造成大量重复工作，而且开发的质量参差不齐，安全性不高。ASP.NET 2.0 的推出，解决了以上问题。ASP.NET 2.0 使用 Membership API 实现与成员验证相关的所有工作，它包括完整的类族和许多使用方便的控件。通过这些控件，能够非常方便地实现用户的注册、验证、信息修改等基本功能。同时，Membership API 还提供了良好的扩展性，以实现用户的特殊需求。

在本项目中将实现的主要功能是用户注册管理功能，具体来说包括以下工作：

① 配置数据库。

② 配置网站安全特性。

③ 创建用户注册页面。

④ 创建用户登录页面。

⑤ 创建显示用户信息页面。

6.3 相关知识

在 ASP.NET 应用程序中，身份验证可以通过以下 4 种形式：

① Windows 身份验证(Windows Authentication)。

② 页面身份验证(Forms Authentication)。

③ 通行证身份验证(Passport Authentication)。

④ 自定义身份验证过程(Acustom Authentication Process)。

绝大多数商业网站都使用页面身份验证，因为这种形式的验证既能适合大多数商业网站的应用需求，又使开发人员能完全控制验证过程，而且不需额外付费。

如要采用页面身份验证，需在 web.config 文件中做相应的配置，最简单的配置如下所示：

```
<system.web>
        <authentication mode = "Forms"/>
</system.web>
```

在上面的配置文件中，仅仅设置了验证模式为页面身份验证，其他的属性都是采用默认值。在默认模式下，用户数据被保存到 App_Data 目录下的一个数据库文件 ASPNETDB.mdf 中，采用 SQL Server 2005 Express 连接模式。如果这种默认设置不符合我们的需要，可以对它进行进一步的设置，下面将详细介绍。

那么，采用了页面身份验证的 ASP.NET 网站，其验证工作机理是怎样的呢？图 6-1 展

示了整个验证流程。

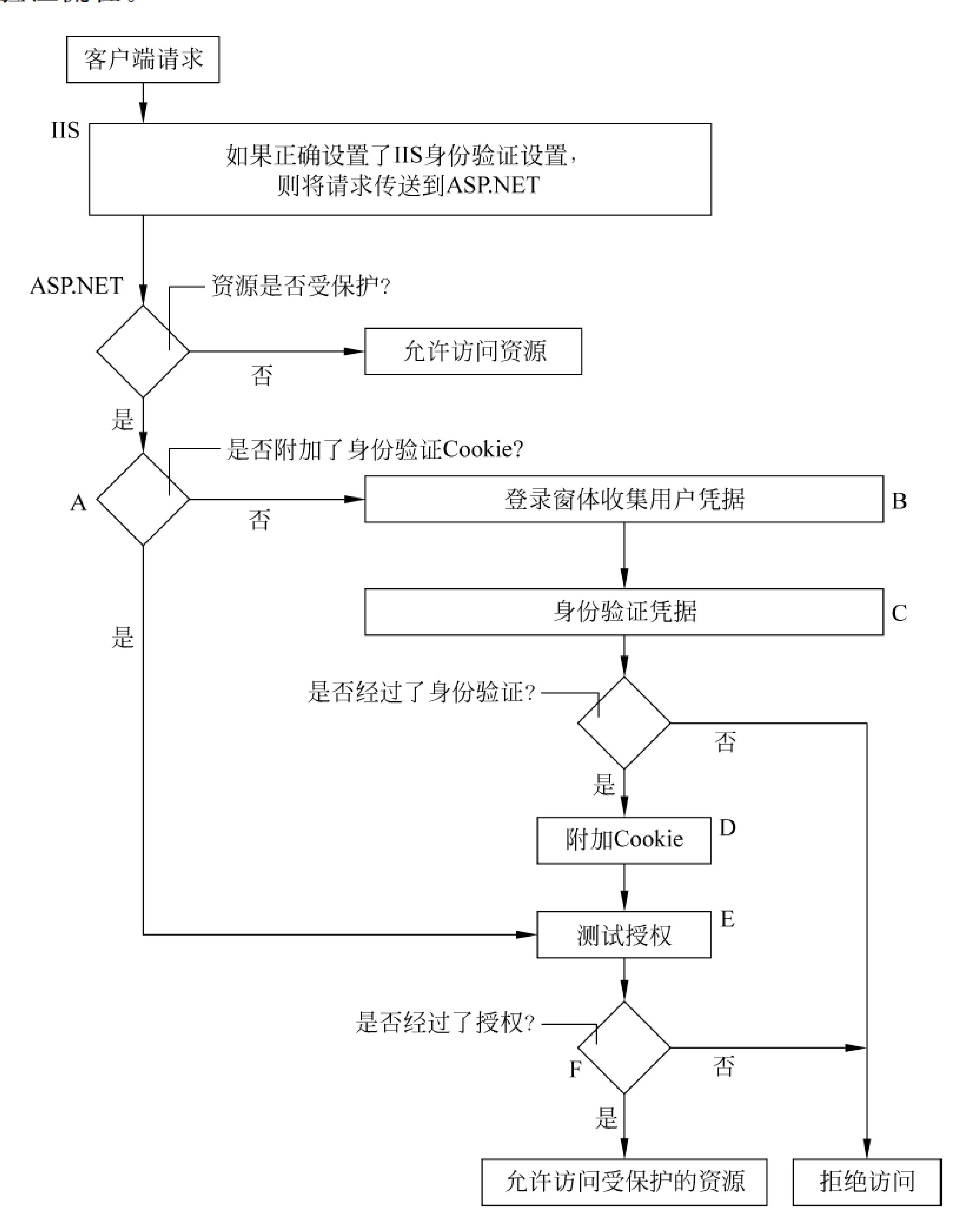


图 6-1　安全性控制流

图 6-1 阐明了当一个请求进入时是如何被处理的。图中标识的 A、B、C 等处是控制流的关键点。

A 点：表示一个请求从 IIS 中导入到一个 ASP.NET 程序中。ASP.NET 程序做的第一件事情就是根据在配置文件中设置的相应信息来判断这个导入的请求是否要求访问一个受限资源。如果这个请求试图访问一个受限资源，那么 ASP.NET 就会检测这个请求是否附加了一个身份验证凭据（ticket）。如果没有，那么该请求会被导向到登录页面（也是在配置文件中定义的）。

犹如去看电影一样，如果没有票就想进入电影院，会发生什么？只能去售票窗口买票，而如果有票那当然是没有问题的。

身份验证凭据（ticket）一般被保存在 Cookie 中，它相当于电影票一样。当用户向服务器发出一个 Http 请求，并且 Cookie 中存在相应的凭据（ticket）时，它就会被附加到请求的

末尾。

A 点做的事情有两件：

① 获得请求并根据在 web.config 文件的<authorization>配置节的配置来判断当前的请求是否是在请求一个受限资源。如果是，就执行第②点；如果不是，则导向登录页面。

② 如果当前的请求是指向一个附带了一个身份验证凭据(ticket)，则可以访问这个资源，否则就会被导向到登录页面(B 点)。

B 点：当一个请求并没有附带一个有效的身份验证凭据时，就会导入到登录页面(B 点)，这个登录页面是我们在<authentication>的<Forms>节中预设的。

登录页面会收集用户提供的身份信息(一般是用户名或密码)，通过对提供的信息进行验证以确定是否在原始的请求上附加一个身份验证凭据，然后访问原始请求所请求的资源(也就是原本应该得到的页面)。

就原始的请求而言，用户访问的是登录页面，而有可能登录页面不是用户最开始请求的。最开始请求的有可能是 default.aspx 页面。

C 点：在收集用户的身份信息后就可以根据它确定是否给予当前请求一个合法的凭证，比如可以把收集到的用户名和密码与数据库中的用户表的信息做比较以确定是否给予凭证。给予用户凭证和验证是否给予凭证的过程完全是两回事。犹如我可以花钱买票也有可能别人送我票，当对于 ASP.NET 而言只要你有合法的凭证就可以访问你想要访问的资源。

D 点：将给予的合法凭证保存到客户端，一般是放在 Cookie 中，在此后客户端发出的新请求中，将自动附上这个凭证。

E 点：这个点负责读取配置文件中相关的授权配置并与身份验证凭证做比较，以确定用户是否可以访问资源。不是已经获得了凭证吗？还需要比较什么？在这里之所以要对比授权的信息是因为虽然获得了合法的凭证但并不表示这个凭证就可以访问任何资源了。在复杂的 ASP.NET 程序中，资源可能会被划分为很多的部分，有些部分不是一般合法用户可以访问的，也就是说需要更高的权限才能访问。E 点所确定的就是当前获得合法凭证的用户到底有没有权利访问他所请求的资源。

F 点：如果当前客户有足够的权限，就将请求的资源发回客户端，否则，拒绝他的访问。

总结一下，对于 ASP.NET 的安全机制而言，会经历以下几个过程：

① 安全机制会拦截所有的进入请求并试图判读这个请求是否附带了身份验证凭据。在这个阶段安全机制会去读取 web.config 以便确定该如何对进入的请求进行判断。

② 没有附加身份验证凭据的请求会被重新导向到登录页面以便完成身份验证的过程。在个这过程中安全机制会去读取 web.config 以便确定该使用什么方式来验证用户。

③ 如果身份验证合法，还需要对获得凭证进行验证，看这个凭证是否有足够的权利访问相应的资源。

④ 以上如果顺利，那么还需要重定向到原始请求的资源上。在这个阶段安全机制也会读取 web.config。

为了正确实现 ASP.NET 的安全机制，一般要完成如下 5 个步骤：

① 配置 web.config 文件，确立适用 Forms 身份验证模式，并且拒绝匿名用户的访问。

② 建立了成员数据存储机制。如果使用的是 SQL Server，则必须建立包含一些表和存

储过程的 SQL Server 数据库。这个过程可由 Visual Studio 2005 工具自动实现。

③ 配置 web.config 文件,确立数据库连接字符串和成员提供程序。

④ 创建一些合法的用户。

⑤ 创建一个登录页,使用系统提供的控件实现登录验证机制。

6.4　项 目 实 施

6.4.1　任务 6-1　配置数据库

1. 任务介绍

对于绝大多数商业网站,用户的注册和验证的核心就是对数据库的访问。当一个用户注册时,也就是将一些经过组织的用户数据保存到指定的数据库。当用户登录时,就是根据用户名和密码在数据库中查找合法的记录,如果找到,则登录成功,否则就是登录失败。本任务就是创建保存用户注册信息的数据库。

2. 任务分析

大多数商业网站在创建时已有一个业务数据库,如果网站不是很大,一般希望将用户注册信息也保存在这个业务数据库中,Visual Studio 2005 提供了专门工具来实现这个需求。所以本任务的核心是通过配置向导工具自动创建和配置用来存放用户信息的 SQL Server 数据库。

(1) 使用向导配置数据库的用户信息

单击“所有程序”→“Microsoft Visual Studio 2005”→“Visual Studio Tools”→“Visual Studio 2005 命令提示”,打开命令行窗体,输入数据库配置命令“aspnet_regsql”,按 Enter 键,出现“ASP.NET SQL Server 安装向导”界面,如图 6-2 所示。

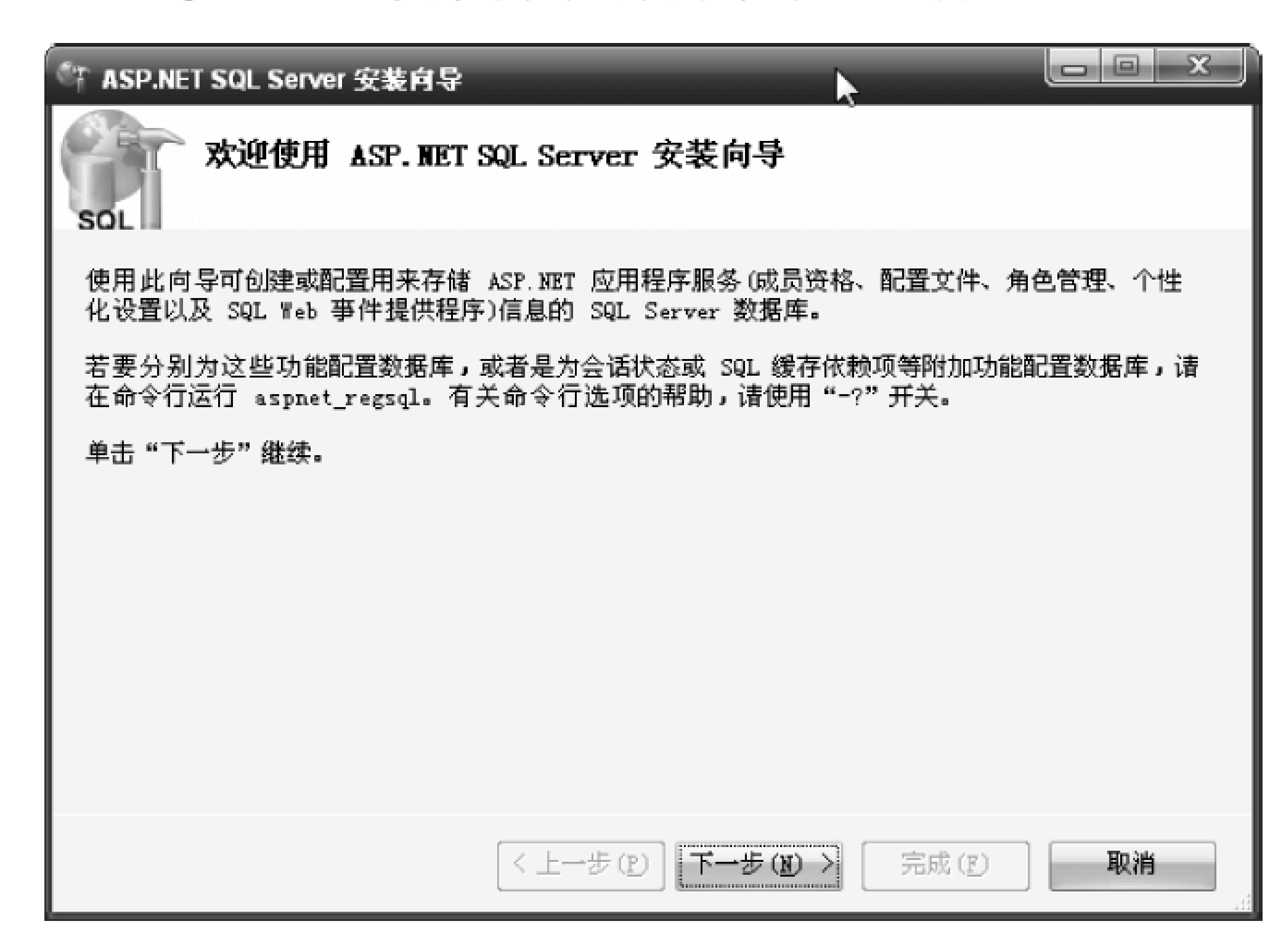


图 6-2　“ASP.NET SQL Server 安装向导”窗口

这个向导能够帮助我们非常方便地创建、配置、修改和删除 SQL Server 数据库中用于存放用户信息的表和存储过程。现在要在已有数据库 NetBook 中创建相关的表和存储过程，单击“下一步”按钮，出现“选择安装选项”界面，如图 6-3 所示。

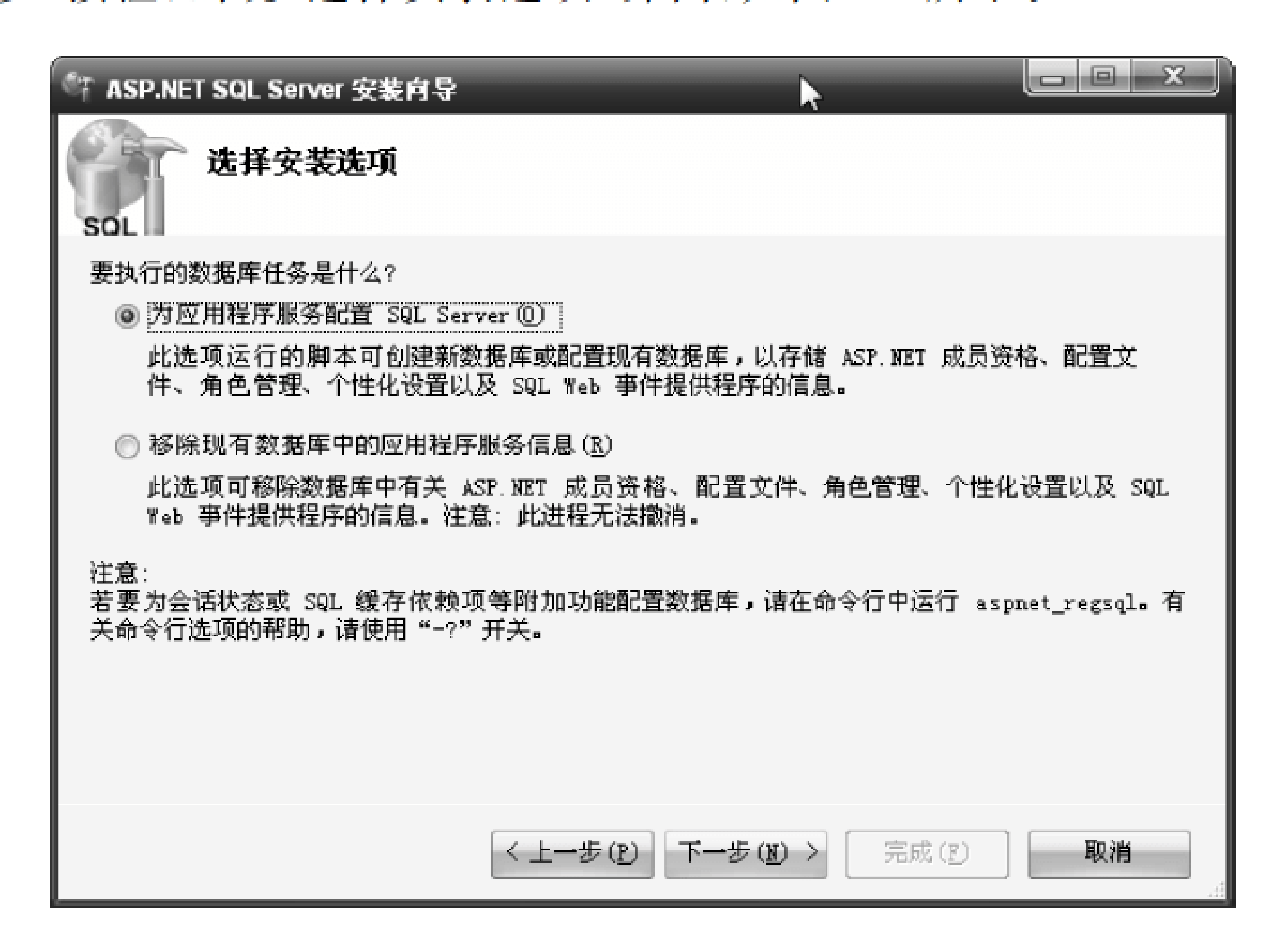


图 6-3 “选择安装选项”窗口

选中“为应用程序服务配置 SQL Server”选项，单击“下一步”按钮，出现“选择服务器和数据库”界面，如图 6-4 所示。

ASP.NET SQL Server 安装向导
选择服务器和数据库
SQL
指定 SQL Server 名称、要创建或移除的数据库的名称，以及连接数据库时要使用的凭据。
注意： 凭据必须标识一个具有创建或移除数据库的权限的用户帐户。
服务器(S): 60AA95CDC5A7467
Windows 身份验证(W)
SQL Server 身份验证(Q)
用户名(U):
密码(A):
数据库(D): Netbook
< 上一步(P)
下一步(N) >
完成(F)
取消

图 6-4 “选择服务器和数据库”窗口

选中我们当前使用的服务器名，在数据库下拉列表中选中要配置的数据库名“NetBook”，单击“下一步”按钮，出现“请确认您的配置”界面，如图 6-5 所示。

确认配置信息无误后，单击“下一步”按钮，出现“数据库已被创建或修改。”界面，如图 6-6 所示。

单击“完成”按钮，结束配置工作。



图 6-5 确认配置



图 6-6 “数据库已被创建或修改。”窗口

(2) 查看数据库的用户信息

当完成步骤(1)的配置工作后，就可以打开数据库 NetBook，查看配置工具对数据库所作的修改。首先可以看到配置工具自动增加了 11 个表，如图 6-7 所示，这些表都以 aspnet_开头，后面的单词表示了表的主要内容。

用户信息主要保存在表 aspnet_Users 和 aspnet_Membership 中，在这些表中存放了 ASP.NET 安全机制提供的默认信息，如用户名、密码、E-mail 地址等，如果还有额外需要存储的信息，例如用户邮寄地址，则需要另外建表来存放。

配置工具还自动添加了很多存储过程，如图 6-8 所示，这些存储过程也以 aspnet_开头，后面的单词表示了存储过程的内容。

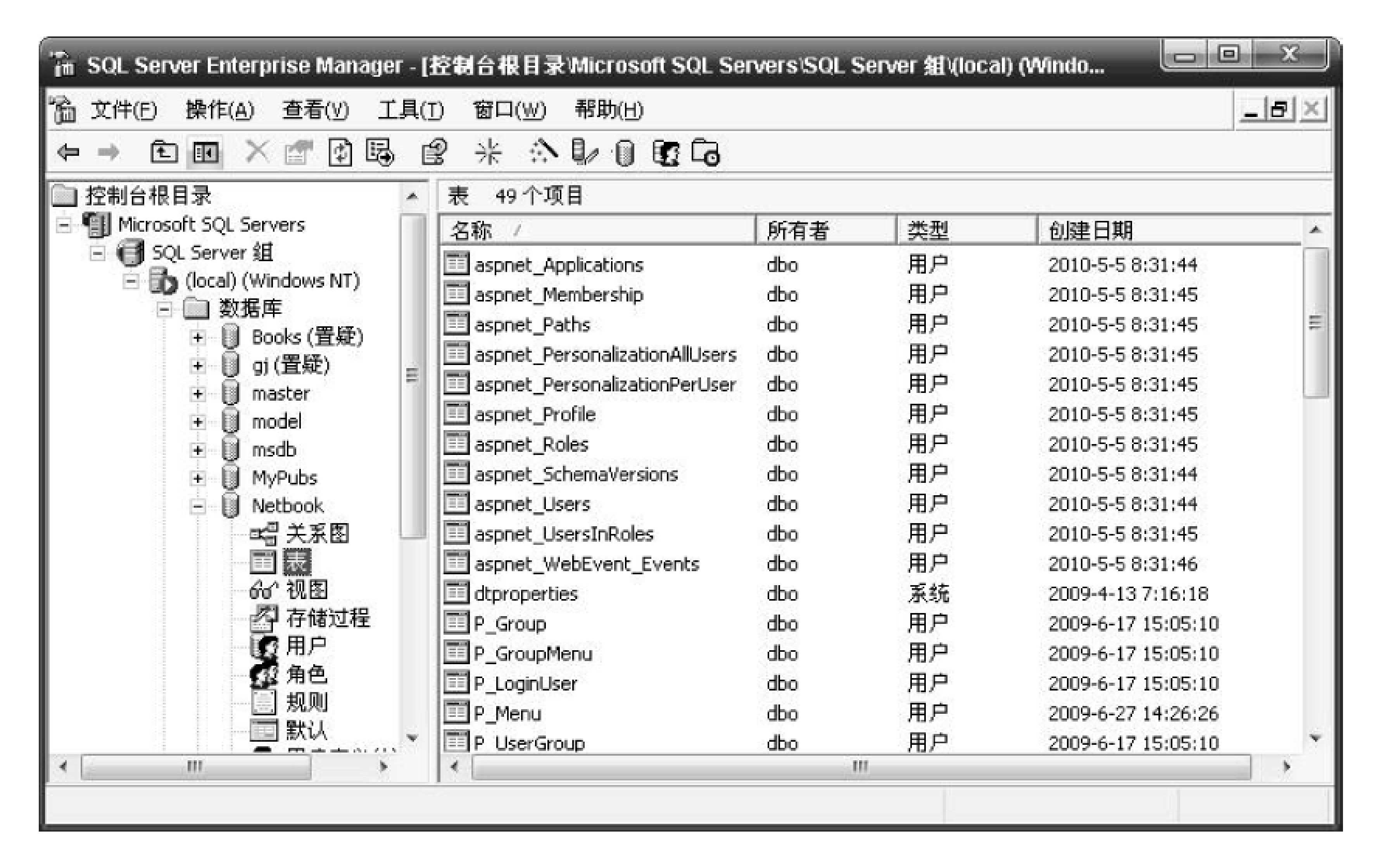


图 6-7　配置工具自动增加的表

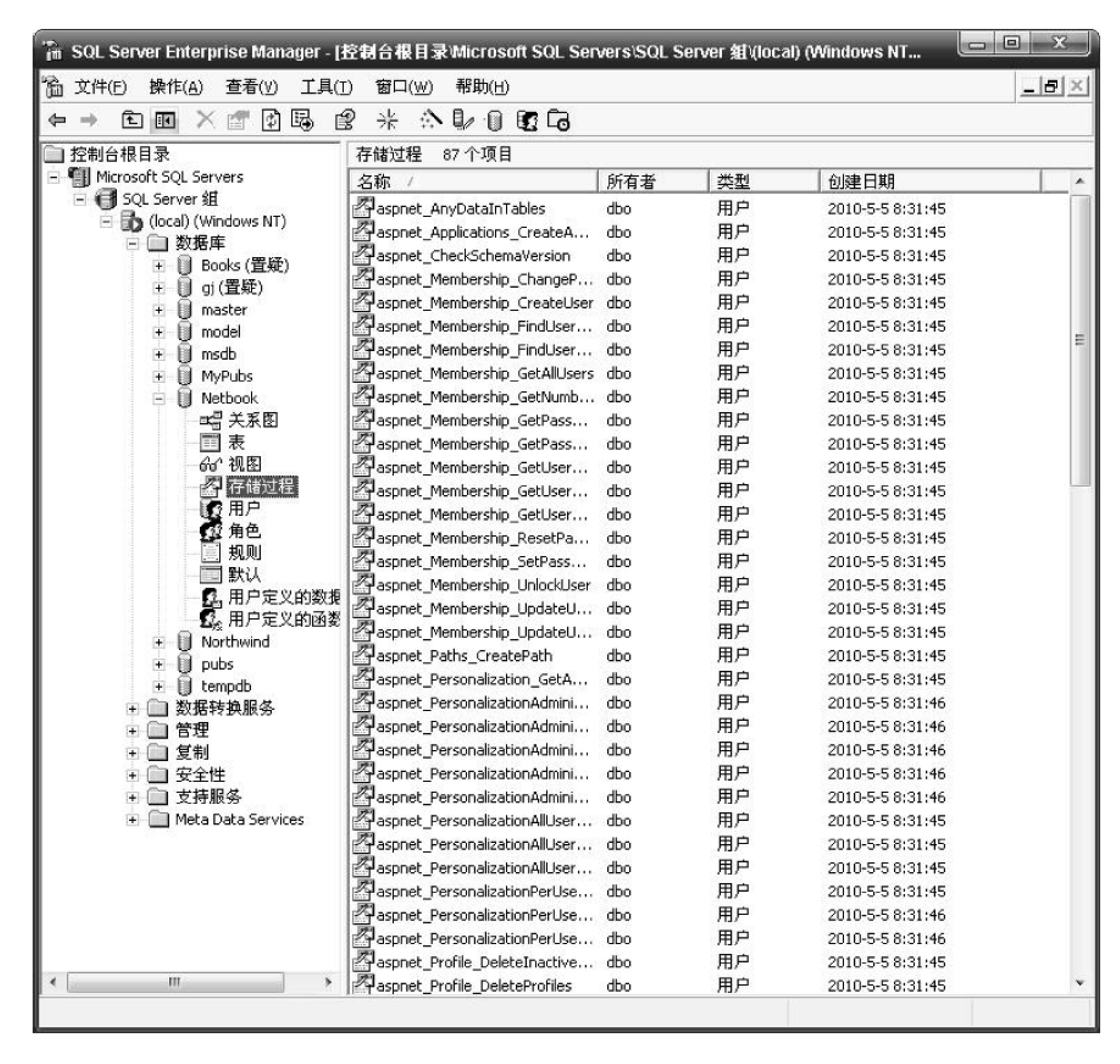


图 6-8　配置工具自动增加的存储过程

3. 任务完成总结

Visual Studio 2005 提供了功能强大的配置工具，以实现自动创建保存用户注册等相关信息的表和存储过程，这些表和存储过程既可以放在一个新创建的数据库中，也可以放到用户已有的业务数据库中。

4. 课堂训练与知识拓展

经过配置向导工具自动创建和配置 SQL Server 数据库，增添了很多表用来存放用户的信息，也增添了很多存储过程来操作这些表。表的重要字段都有严谨的约束，表之间都有良好的关系定义，熟悉这些表和存储过程对使用 ASP.NET 的核心安全机制有良好的帮助。

6.4.2 任务 6-2 配置 Web 程序

1. 任务介绍

ASP.NET 安全设置是在 machine.config 和 web.config 文件中进行配置的。在当前 .NET Framework 安装目录下的 config 子目录中的 machine.config 文件，建立了基本设置和默认设置。如果要对某一 Web 程序进行配置，比如要进行页面认证，就必须修改它的 web.config 文件，在其中建立特定的设置。

本任务的目的是配置 Web 程序，也就是修改 web.config 文件，使得网站能够采用合适的方法来进行安全验证。

2. 任务分析

本任务的目的是配置 Web 程序，使得网站能够采用合适的方法来进行安全验证，这就必须修改 web.config 文件。我们既可以手工编辑 web.config 文件，也可以使用网站管理工具来配置 Web 程序，配置的结果就是自动地修改 web.config 文件。采用网站管理工具能够更直观、更方便地对 Web 程序的主要方面做出配置。

(1) 配置 web.config 文件

ASP.NET 安全设置是在 machine.config 和 web.config 文件中进行配置的。在当前 .NET Framework 安装目录下的 config 子目录中的 machine.config 文件，建立了基本设置和默认设置。可以在网站根目录和应用程序根目录中的 web.config 文件中建立特定于站点的设置和特定于应用程序的设置（包括重写 machine.config 文件中的设置）。子目录会继承上一级目录的设置，除非子目录中的 web.config 文件重写这些设置。

web.config 文件有 3 个主要的子节：authentication，authorization 和 identity 节。通常在 machine.config 文件中设置各个安全元素的值，并根据需要在应用程序级别的 web.config 文件中重写这些值。所有子目录都自动继承那些设置，但是，子目录可以有自己的配置文件，这些配置文件重写继承的设置。

打开网站的 web.config 文件，添入如下程序中带灰底纹的语句行。

```
<?xml version="1.0"?>
<!--
    注意: 除了手动编辑此文件以外,还可以使用
    Web 管理工具来配置应用程序的设置.可以使用 Visual Studio 中的
     "网站"->"Asp.NET 配置"选项.
    设置和注释的完整列表在
    machine.config.comments 中,该文件通常位于
    \Windows\Microsoft.NET\Framework\v2.x\Config 中
 -->
<configuration xmlns="http://schemas.microsoft.com/.NETConfiguration/v2.0">
    <appSettings/>
    <connectionStrings>
  <add name="NetBookConnectionString" connectionString="Data Source=127.0.0.1;Initial
Catalog=NetBook;Integrated Security=True"
    providerName="System.Data.SqlClient" />

  </connectionStrings>
      <system.web>
          <!--
              设置 compilation debug="true" 将调试符号插入
              已编译的页面中.但由于这会
              影响性能,因此只在开发过程中将此值
              设置为 True.
          -->
          <compilation debug="true"/>
          <!--
              通过 <authentication> 节可以配置 ASP.NET 使用的
              安全身份验证模式,
              以标识传入的用户.
          -->
          <authentication mode="Windows"/>
      <membership defaultProvider="MyMembershipProvider">
        <providers>
          <add name="MyMembershipProvider"
               connectionStringName="NetBookConnectionString"
               applicationName="MyMembership"
               enablePasswordRetrieval="false"
               enablePasswordReset="true"
               requiresQuestionAndAnswer="true"
               requiresUniqueEmail="true"
               passwordFormat="Clear"
               type="System.Web.Security.SqlMembershipProvider" />
        </providers>
      </membership>
          <!--
              如果在执行请求的过程中出现未处理的错误,
              则通过 <customErrors> 节可以配置相应的处理步骤.具体说来,
              开发人员通过该节可以配置
              要显示的 html 错误页
              以代替错误堆栈跟踪.
```

```
        <customErrors mode = "RemoteOnly" defaultRedirect = "GenericErrorPage.htm">
            <error statusCode = "403" redirect = "NoAccess.htm" />
            <error statusCode = "404" redirect = "FileNotFound.htm" />
        </customErrors>
        -->
    </system.web>
</configuration>
```

在默认配置下，用户数据被保存到 App_Data 目录下的一个数据库文件 ASPNETDB.mdf 中，采用 SQL Server 2005 Express 连接模式。而现在要使用 SQL Server 2000 服务器中的数据库 NetBook，故重新配置了用来提供数据库读写服务的提供程序。在这添加的一段中，定义了要使用的数据库连接字符串 NetBookConnectionString，而这个连接字符串指向了数据库 NetBook。段中其他的配置与默认配置相同，用来确定对成员身份验证的一些细节。

(2) 使用网站管理工具配置 Web 程序

虽然我们可以通过编辑 web.config 文件的方法来实现对网站的配置，但毕竟比较繁琐，ASP.NET 2.0 还提供了另一种方便实用的配置工具，即通过图形界面的管理工具来进行配置。

首先，单击 Visual Studio 2005 的"网站"菜单，单击"ASP.NET 配置"菜单项，在浏览器中弹出如图 6-9 所示的配置管理工具界面。

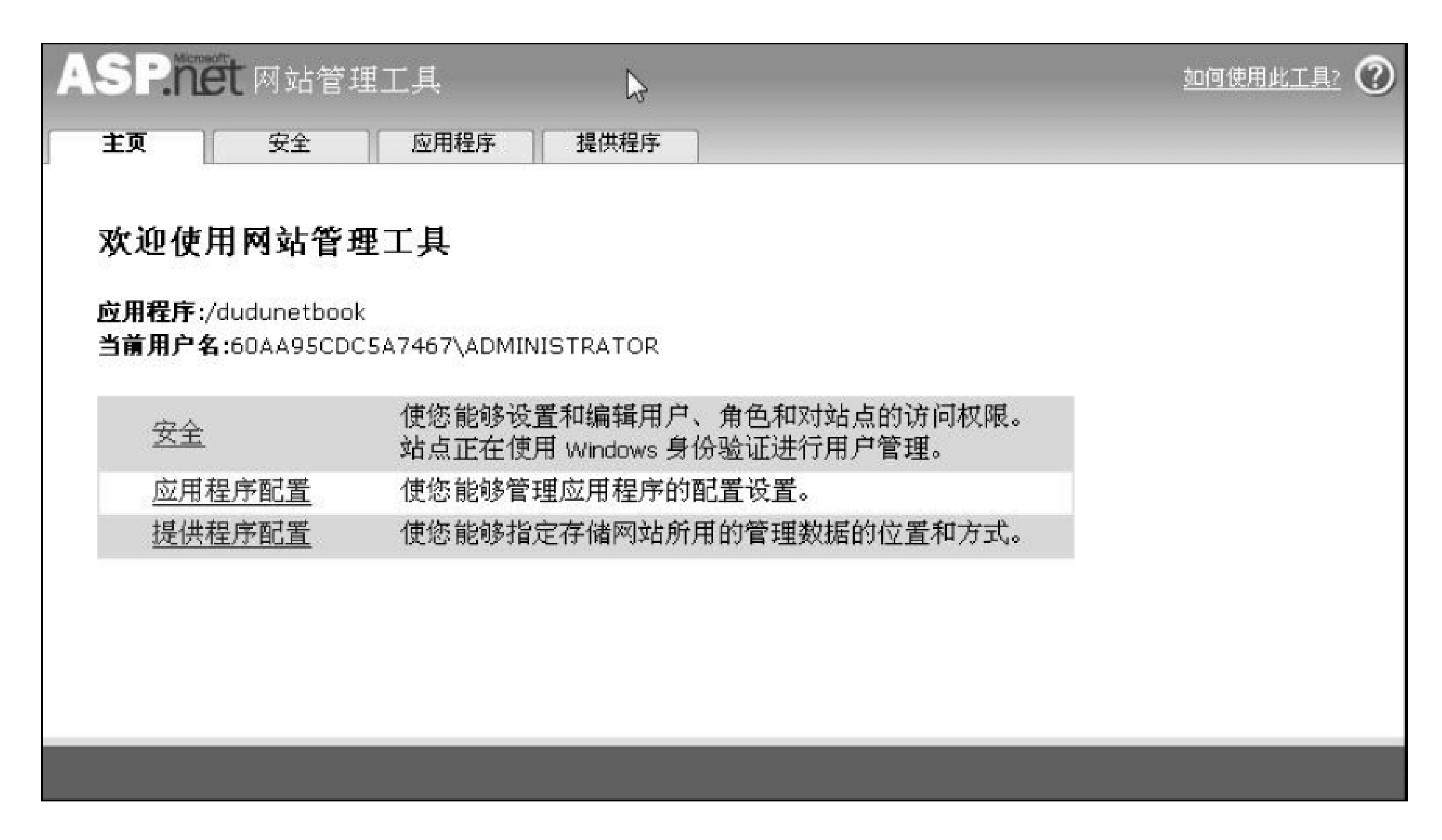


图 6-9　网站配置管理工具

在这个配置管理工具中，有一个选项卡式界面，该界面在下列选项卡上对相关的配置设置进行分组：

- "安全"选项卡，其中包含有助于保护 Web 应用程序资源并管理用户账户和角色的设置。
- "主页"选项卡，提供对本配置管理工具的简要说明和其他选项卡的描述。
- "应用程序"选项卡，其中包含用来管理影响 ASP.NET 应用程序的配置元素的设置。

- “提供程序”选项卡，其中包含用来添加、编辑、删除、测试或分配应用程序提供程序的设置。

单击“安全”选项卡，出现如图6-10所示的安全配置界面。



图 6-10 默认安全配置

从“用户”一栏可以看到，当前系统默认的身份验证类型是Windows，为了改成需要的Forms验证方式，单击“选择身份验证类型”文字链接，出现如图6-11所示的身份验证类型配置界面。

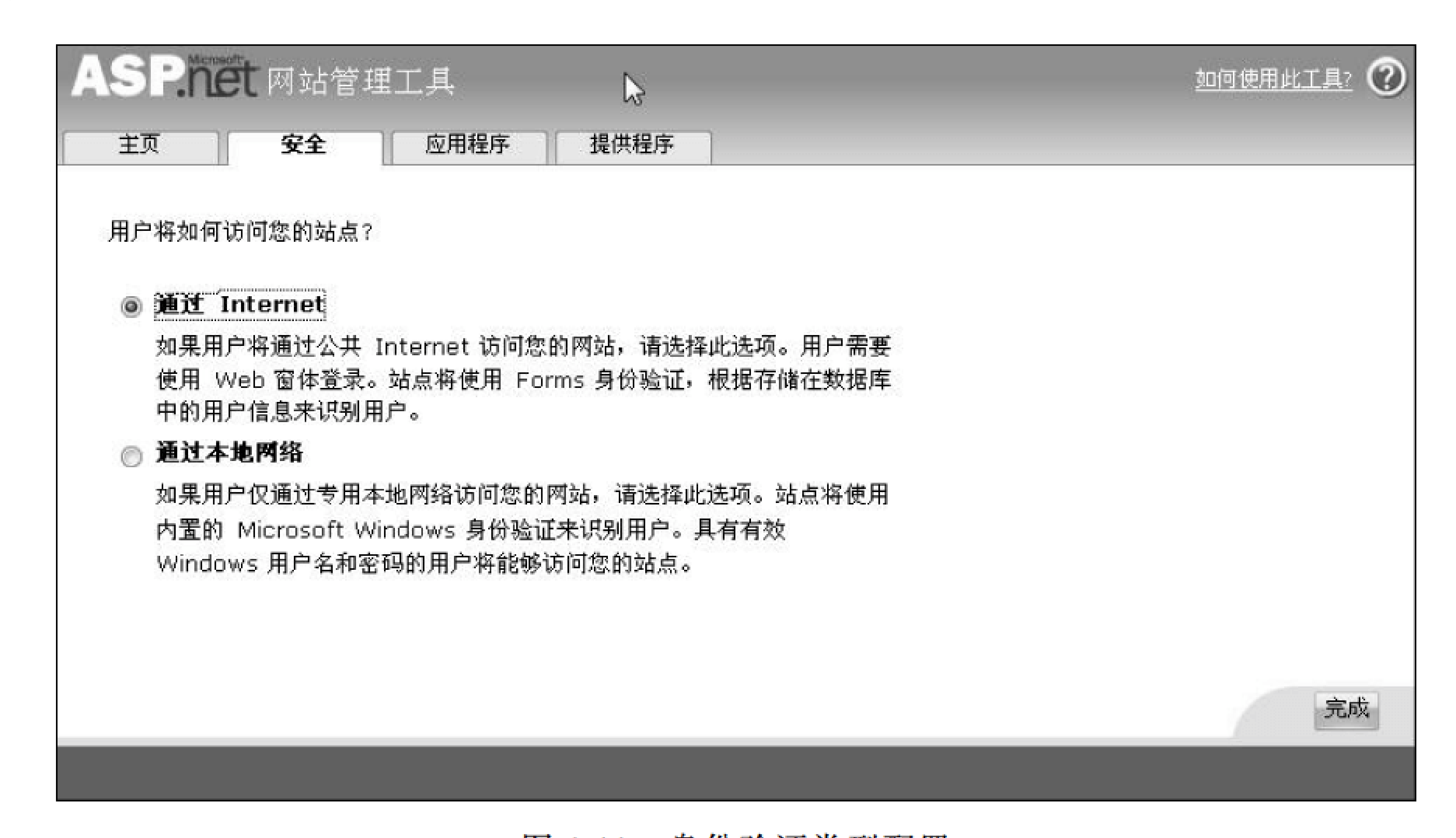


图 6-11 身份验证类型配置

选中“通过Internet”选项，单击“完成”按钮，出现如图6-12所示的使用Forms验证方式的安全配置界面。

从用户界面可以看到，目前的注册用户数目为0，单击“创建用户”，出现如图6-13所示的创建用户界面。





图 6-12　使用 Forms 验证方式的安全配置

ASP.net 网站管理工具　如何使用此工具?

主页　安全　应用程序　提供程序

通过在本页上输入用户的 ID、密码和电子邮件地址来添加用户。

创建用户

注册新账户

用户名: testOne

密码: ●●●●●●●●●

确认密码: ●●●●●●●●●

电子邮件: one@one.com

安全提示问题: hello

安全答案: one

创建用户

角色

未启用角色。

☑活动用户

上一步

图 6-13　创建用户

作为测试，按照图 6-13 所示填入内容，需要注意的是，根据默认的密码要求，必须至少 7 位并且至少一位是非字母和数字的，合法的密码形如“hello_one”、“user1_test”等。输入完成后单击“创建用户”按钮，出现如图 6-14 所示的完成用户创建提示界面。

单击“上一步”按钮，回到如图 6-15 所示的具有 1 个用户的安全配置界面。

在这个界面中不仅可以知道当前用户的数目，也可以对用户信息进行修改和删除。单



图 6-14　完成用户创建提示



图 6-15　具有 1 个用户的安全配置

击“管理用户”，出现如图 6-16 所示的管理用户界面。

在用户列表中选中要修改的用户，比如“testOne”，单击“编辑用户”，出现如图 6-17 所示的编辑用户界面。

编辑完成后，单击“保存”按钮，就回到如图 6-10 所示的“安全配置”界面，在安全配置界面上单击“创建访问规则”，出现如图 6-18 所示的创建访问规则界面。



图 6-16　管理用户

图 6-17　编辑用户

现在要禁止匿名用户访问网站 manage 子目录下的所有文件。首先在“添加新访问规则”表的左侧选中子目录“manage”，再勾选右侧“匿名用户”和“拒绝”选项，单击“确定”按钮，将创建一条访问规则，禁止所有匿名用户访问子目录 manage 内文件。

在图 6-10 所示的“安全配置”界面上，单击“提供程序”选项卡，出现如图 6-19 所示的“提供程序配置”界面。



图 6-18　创建访问规则

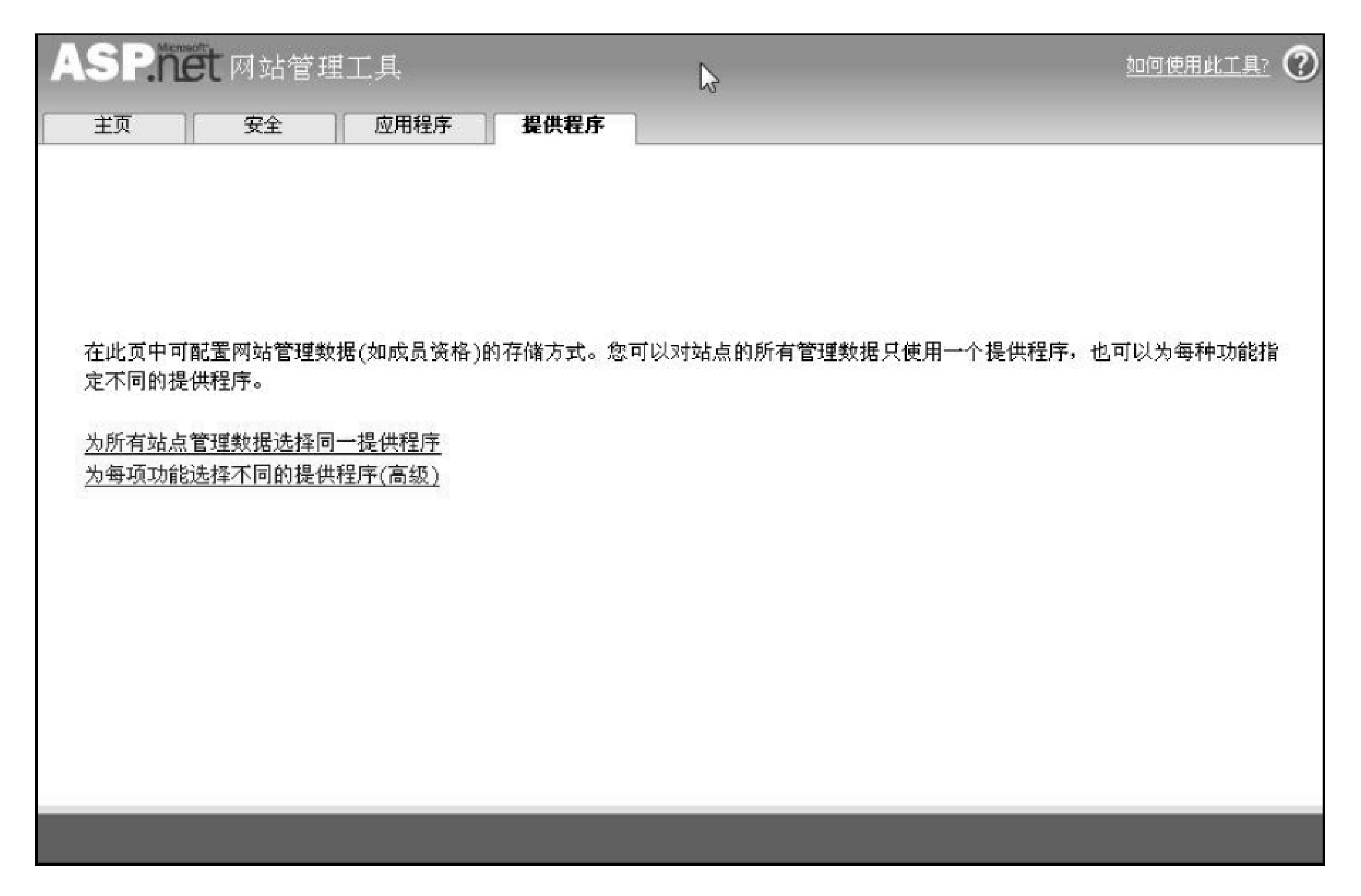


图 6-19　提供程序配置

提供程序(Provider)是 ASP.NET 提供身份验证和角色服务的一系列类和实例对象的总称,它们有默认的工作规则,可以通过配置进行修改。我们在步骤(1)已经修改过 web.config 文件,采用自定义的身份验证提供程序,现在就可以检查一下。单击“为每项功能选择不同的提供程序(高级)”文字链接,出现如图 6-20 所示的提供程序配置界面。

身份验证服务默认的提供程序称为“AspNetMembershipProvider”,确认选中的是在步骤(1)中设置的“MyMembershipProvider”,单击“上一步”按钮,回到提供程序界面,结束

图 6-20　提供程序配置

配置。

打开 web.config 文件，会发现如下一行发生了修改（它表示采用 Forms 身份验证方式）：

```
<authentication mode = "Forms" />
```

并且在 manage 子目录下新增了一个 web.config 文件，内容如下所示：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 注意：除了手动编辑此文件以外，还可以使用 Web 管理工具来
     配置应用程序的设置.可以使用 Visual Studio 中的"网站" ->"ASP.NET 配置"
     选项.
     设置和注释的完整列表在 machine.config.comments 中，
     该文件通常位于 \Windows\Microsoft.NET\Framework\v2.0.xxxxx\Config 中
 -->
<configuration>
     <appSettings/>
     <connectionStrings/>
     <system.web>
          <authorization>
               <deny users = "?" />
          </authorization>
     </system.web>
</configuration>
```

它表示在当前目录下（也即是 manage 子目录下）禁止匿名用户访问。在配置文件中“?”表示匿名用户，“*”表示所有用户。

这说明通过网站管理工具所作的任何修改，都将在 web.config 文件中自动得到反映。网站管理工具其实就是通过修改 web.config 文件来实现对网站的配置。

3. 任务完成总结

网站的安全验证有多种形式，本任务的核心工作就是配置 Web 程序，使得网站能够采用合适的方法进行安全验证，这必须修改 web.config 文件。既可以手工编辑 web.config 文件，也可以使用网站管理工具来配置 Web 程序，这种方式更直观、更方便。采用网站管理工配置的结果会自动地修改 web.config 文件。

4. 课堂训练与知识拓展

web.config 文件有 3 个主要的子节：authentication，authorization 和 identity 节。通常在 machine.config 文件中设置各个安全元素的值，并根据需要在应用程序级别的 web.config 文件中重写这些值。所有子目录都自动继承那些设置，但是，子目录可以有自己的配置文件，这些配置文件重写继承的设置。

理解这些配置定义可以大大提高我们对 Web 程序的控制能力，详细而严谨的定义可以查阅微软的官方文档。

6.4.3 任务 6-3 使用登录类控件实现用户登录注册功能

1. 任务介绍

当用户注册时，必须有一个界面，让用户输入用户名、密码、E-mail 等相关信息，系统检验合法后把它们存到数据库中，判定注册成功。

当用户登录时，也必须有一个界面，让用户输入用户名和密码，系统根据用户名和密码在数据库中查找合法的记录，如果找到，则登录成功，否则登录失败。本任务就是创建完成注册和登录工作的页面。通过使用 ASP.NET 2.0 的相关控件，整个创建工作很容易，而读取数据库和验证工作由控件联合一些集成的类族自动完成。

2. 任务分析

对于绝大多数商业网站，用户的注册和验证，其核心就是对数据库的访问。当一个用户注册时，也就是将一些经过组织的用户数据保存到指定的数据库。当用户登录时，就是根据用户名和密码在数据库中查找合法的记录，如果找到，则登录成功，否则登录失败。

ASP.NET 2.0 使用 Membership API 来实现与成员验证相关的所有工作，它包括完整的类族和许多使用方便的控件。Login 控件和 CreateUserWizard 控件就是其中的典型。通过使用这些控件，能够非常方便地实现用户的注册、验证、登录等基本功能。同时，还可以设置修改 CreateUserWizard 控件的默认行为模式，插入邮寄地址等特殊信息。

(1) 使用 Login 控件实现用户登录功能

在任务 6-2 中我们已经对网站的安全做了新的配置，禁止匿名用户访问网站 manage 子目录下所有文件，如果用户请求了 manage 子目录下的页面，浏览器将重定向到如图 6-21 所示的错误页。

造成这个错误的原因是客户端请求了 manage 子目录下的资源，而根据安全配置是不



图 6-21　HTTP 404 错误页

允许匿名访问的，所以系统就自动重定向到登录页，默认的登录页名称是 Login.aspx，下面就来创建它。

首先添加一个新 Web 窗体，命名为 Login.aspx。从工具箱的“登录”面板中拖放一个 Login 控件和一个 LinkButton 控件到页面中，如图 6-22 所示。

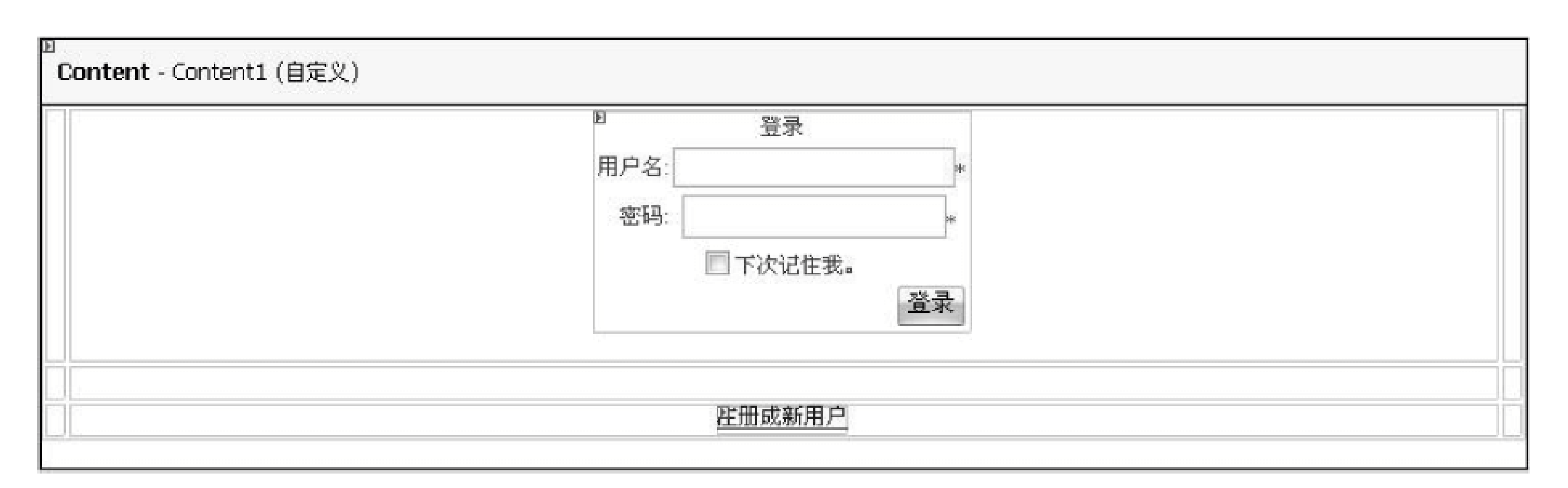


图 6-22　登录页面

运行后出现如图 6-23 所示界面，我们输入在任务 6-2 中创建的用户信息，单击“登录”按钮就完成了登录过程。

Login 控件是个非常方便的工具，默认状态下，它能完成几乎所有有关登录的功能，诸如让用户输入信息、验证信息格式、访问数据库、判断登录是否成功等。如果还需进一步扩展功能，可以单击它的智能按钮，弹出如图 6-24 所示的菜单。“自动套用格式…”菜单项能够非常方便地设置登录控件的外观，“转换为模板”菜单项能够使登录控件转换为一个模板，在这个模板中可以编辑和修改界面中的文本框、复选框等各个元素，还可以增加额外的元素。



图 6-23 实现登录过程



图 6-24 Login 控件的智能菜单项

(2) 使用 CreateUserWizard 控件实现用户注册功能

首先添加一个新 Web 窗体,命名为 register.aspx。从工具箱的"登录"面板中拖放一个 CreateUserWizard 控件到页面中,如图 6-25 所示。

单击 CreateUserWizard 控件的智能按钮,弹出动态菜单,包含了对控件可做的全部操作。从"步骤"列表框可以看出整个用户创建过程分"注册新账户"和"完成"两个步骤,初始打开的"注册新账户"步骤,列出了许多文本输入框和格式验证控件,用以输入新用户的信息。当我们单击"步骤"列表框的"完成"项时,CreateUserWizard 控件转到"完成"步骤,如图 6-26 所示。

CreateUserWizard 控件的"注册新账户步骤"提供了一些文本输入框,能满足一般的注册新用户的需求。如果在注册时还需要输入更多的信息,比如真实姓名和实际地址,则可以通过添加新用户创建的步骤或修改"注册新账户步骤"界面来实现。下面就采用第二种方法来添加输入真实姓名和地址的功能。

在 CreateUserWizard 控件智能菜单的步骤列表框上单击"注册新账户"项,然后单击

图 6-25　CreateUserWizard 控件注册新账户步骤界面



图 6-26　CreateUserWizard 控件完成界面

“自定义创建用户步骤”，显示如图 6-27 所示界面。

经过这个转换，CreateUserWizard 控件的“注册新账户”步骤界面变成了一个模板，里面的标签、文本输入框等控件都可以进行编辑。现在仿造上面的格式新增两个标签控件，分别表示真实姓名和邮寄地址，再增添两个文本输入框控件，分别命名为 textboxName 和 textboxAddress。修改后 CreateUserWizard 控件的源代码如下所示：

```
<asp:CreateUserWizard ID = "CreateUserWizard1" runat = "server" OnCreatedUser =
"CreateUserWizard1_CreatedUser">
     <WizardSteps>
          <asp:CreateUserWizardStep ID = "Info" runat = "server">
              <ContentTemplate>
                 <table border = "0">
                        ⋮
                 </table>
              </ContentTemplate>
            </asp:CreateUserWizardStep>
            <asp:CompleteWizardStep ID = "CompleteWizardStep1" runat = "server">
            </asp:CompleteWizardStep>
```

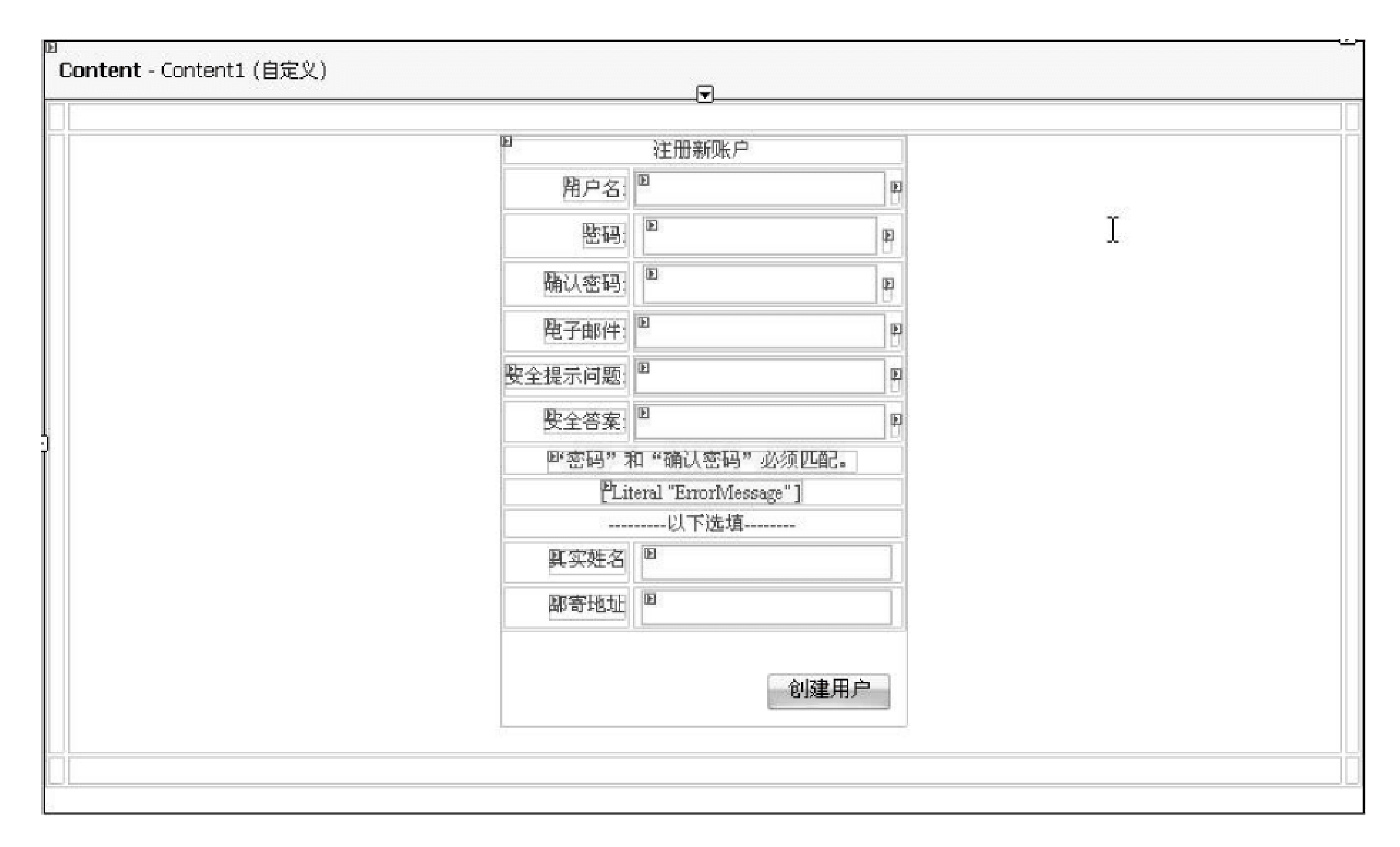


图 6-27　自定义创建用户步骤

```
        </WizardSteps>
    </asp:CreateUserWizard>
```

从以上的源代码可以看出，CreateUserWizard 控件包含一个向导步骤集 WizardSteps，其中包含两个步骤：CreateUserWizardStep 和 CompleteWizardStep，分别对应智能菜单中的“注册新账户”和“完成”步骤。在 CreateUserWizardStep 中又包含一个内容模板，其中存放了多个标签和文本框控件。将 CreateUserWizardStep 的 ID 设置为“Info”，以方便在代码中查阅。

真实姓名和邮寄地址是注册新用户时的选填项，如果用户填了信息，就在数据库的自定义用户表里插入一条记录；如果用户没填，则不对自定义用户表操作。因此，这个操作可以放在创建了一个新用户后即刻进行，比较理想的是放在 CreateUserWizard 控件的 CreatedUser 事件中执行，这个事件在一个新用户被创建后触发，下面是参考代码。

```
protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
{
        WizardStepBase Step = null;
        //遍历创建步骤，找到名为 Info 的那一步，也即是第一步。
        for (int i = 0; i < CreateUserWizard1.WizardSteps.Count; i++)
        {
            if(CreateUserWizard1.WizardSteps[i].ID== "Info")
                step = CreateUserWizard1.WizardSteps[i];
        }
        if (step != null)
        {
            TextBox textBoxName = (TextBox)(step.Controls[0].FindControl("textboxName"));
            TextBox textBoxAddress = (TextBox)(step.Controls[0].FindControl("textboxAddress"));

            string strRealName = textBoxName.Text;
```

```
                string strAddreass = textBoxAddress.Text;

                //如果用户填了信息,就在数据库的自定义用户表里插入一条记录。
                if (strRealName != string.Empty)
                {
                    //以下实现向数据库的自定义用户表插入记录,因篇幅缘故省略。
                }
            }
}
```

在运行时输入信息,单击“创建用户”按钮就能顺利地注册一个新用户,如图6-28所示。



图6-28　注册新用户

3. 任务完成总结

本任务通过使用ASP.NET 2.0的相关控件,创建了注册和登录页面,完成了相关工作。ASP.NET 2.0使用Membership API来实现与成员验证相关的所有工作,它包括完整的类族和许多使用方便的控件。通过使用这些控件,我们能够非常方便地实现用户的注册、验证、登录等基本功能,还可以通过设置修改CreateUserWizard控件的默认行为模式,在注册时收集用户邮寄地址等特殊信息。

4. 课堂训练与知识拓展

CreateUserWizard控件的“注册新账户步骤”能满足一般性的注册需求,但在实际应用中,往往需要输入更多的信息,如真实姓名、地址、个人爱好等,这些内容的输入可以通过添

加新步骤或修改“注册新账户步骤”界面来实现。在步骤(2)中我们介绍了第二种方法，用来添加输入真实姓名和地址。如果有较多的额外信息需要收集，我们可以尝试采用第一种方法，把采集额外信息作为一个单独的步骤。

6.4.4 任务6-4 使用登录类控件实现显示用户状态功能

1. 任务介绍

当用户处于匿名状态时，它需要一个进行登录操作的入口，而在已登录状态时，需要一个退出登录的链接。许多网站还能够显示登录用户的用户名，让用户感到很亲切。在本任务中，我们就来实现这样的功能。

2. 任务分析

ASP.NET 2.0 使用 Membership API 来实现与成员验证相关的所有工作，它包括完整的类族和许多使用方便的控件。通过使用这些控件，我们能够非常方便地实现用户的注册、验证、登录等基本功能。LoginStatus 控件和 LoginName 控件就是其中非常小巧实用的控件，通过它们能非常简便地实现显示用户状态和用户姓名的功能。

(1) 使用 LoginStatus 控件实现登录

在首页上拖放进一个 LoginStatus 控件，如图 6-29 所示。

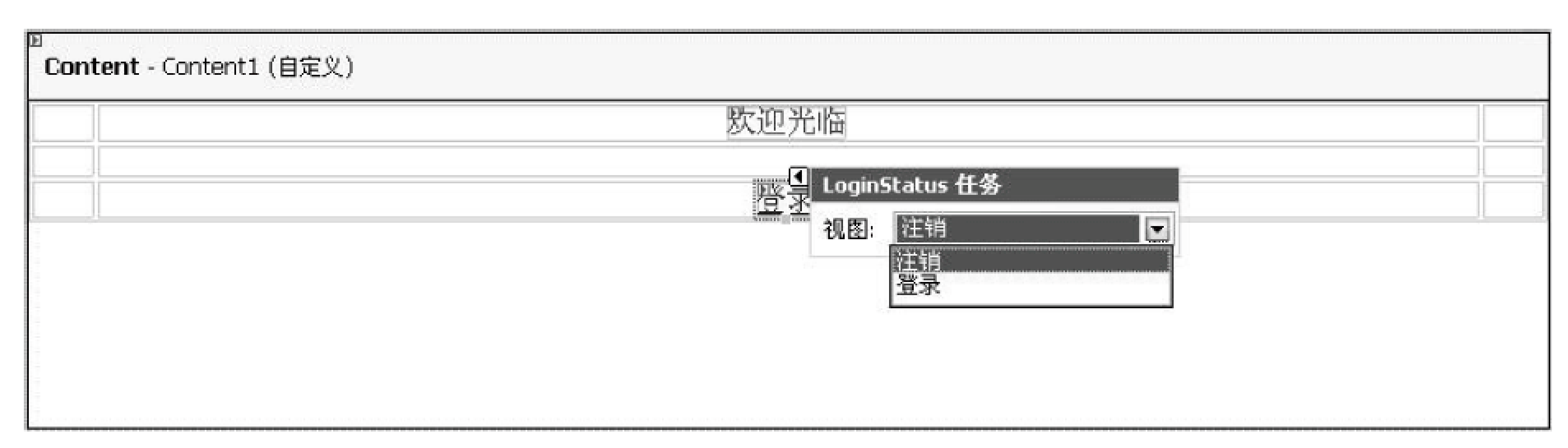


图 6-29　LoginStatus 控件

LoginStatus 控件为没有通过身份验证的用户显示登录链接，为通过身份验证的用户显示注销链接。登录链接将用户带到登录页。注销链接将当前用户的标识重置为匿名用户。

(2) 使用 LoginName 控件显示用户姓名

在首页上拖放进一个 LoginName 控件，如图 6-30 所示。

图 6-30　LoginName 控件

如果用户已使用 ASP.NET 成员身份登录，则 LoginName 控件显示该用户的登录名。如果网站使用集成的 Windows 身份验证，该控件则显示用户的 Windows 账户名。

运行程序，登录后整体效果如图 6-31 所示。



图 6-31 LoginStatus 控件和 LoginName 控件运行效果

3. 任务完成总结

ASP.NET 2.0 提供很多使用方便的控件，使我们能够迅速地实现与用户的注册、登录等相关的功能。LoginStatus 控件能够显示当前用户的状态，并提供登录或退出的链接，LoginName 控件能显示当前用户的姓名，使网站显得更个性化。

4. 课堂训练与知识拓展

注册之后有时可能需要修改密码，ASP.NET 2.0 提供了一个使用非常方便的控件 ChangePassword。创建一个页面，使用这一控件来实现这一功能。

6.5 项 目 总 结

本项目采用数据库配置工具自动配置了身份验证所需的数据库，采用网站管理工具配置了网站的安全设置，最后用一系列登录控件实现了登录、注册等功能。

众多 ASP.NET 登录控件一起为 ASP.NET Web 应用程序提供了可靠完整且不需要编程的登录解决方案。默认情况下，登录控件与 ASP.NET 成员身份集成，从而有助于网站的用户身份验证过程实现自动化。

CreateUserWizard 控件收集潜在用户所提供的信息，创建新的用户。

Login 控件显示供用户输入身份验证信息的用户界面，包含用于用户名和密码的文本框和一个复选框，该复选框可用于指示是否需要服务器使用 ASP.NET 成员身份存储他们的标识并且在他们下次访问该网站时自动进行身份验证。

LoginName 控件显示当用户使用 ASP.NET 成员身份登录时的用户登录名。如果网站使用集成的 Windows 身份验证，该控件则显示用户的 Windows 账户名。

LoginStatus 控件为没有通过身份验证的用户显示登录链接，为已通过身份验证的用户显示注销链接。登录链接将用户带到登录页。注销链接将当前用户的标识重置为匿名用户。

6.6 项目实训

1. 任务描述

创建一个反映出版社信息的网站，采用示例数据库 Pubs，这个网站的部分内容只允许注册用户访问。当用户注册时，除了要提供用户名、密码、E-mail 等常规信息之外，还要提供真实姓名、年龄、手机号码和真实地址等信息。在网站的模板上，有反映用户登录状态以及用户名的信息。

2. 任务要求

① 采用数据库配置工具配置示例数据库 Pubs。

② 采用网站管理工具配置网站的安全设置，manage 子目录下的资源只允许注册用户访问。

③ 将 Login 控件和 LoginStatus 控件用在模板上。

④ 为创建新用户过程增添新的步骤以收集额外信息。

设计网上书店购物车

7.1 项目介绍

网上书店一般都有购物车功能。客户查询到自己喜爱的图书,有购买意向时,便可以将图书放到购物车内,在网站暂存。我们到超市购物时,可以将自己喜爱的商品放入购物车,也可以将购物车中不需要的商品再放回超市的货架。网上购物车功能类似于超市购物车的功能,网上购物车可以实现商品信息的添加和删除,能够实现计算每种商品的小计价格和整个购物车内商品的总价。为吸引客户,网站应提供匿名用户,也可以使用购物车的个性化服务。在项目7中将介绍如何设计网上书店购物车,并实现匿名用户使用购物车,以及用户登录后匿名用户购物车数据向注册用户购物车的迁移。

7.2 项目分析

客户浏览网上书店时,通常会先进入网站的首页浏览网站图书信息,此时客户并没有使用账户登录网站,即用户是以匿名方式访问网站的。当客户看到自己喜爱的图书时,将该图书添加到匿名购物车内保存。客户离开网站,下次登录网站时,客户打开匿名购物车,上次选中的图书会仍然在匿名购物车中,没有丢失。客户结账时,需要登录网站,匿名购物车中的商品信息要能够自动转到客户购物车中。这一个性化服务的购物流程符合客户的操作习惯。传统的购物车设计一般使用Session对象保存购物车对象,当客户只进行了购物操作,没有下订单操作就离开网站时,客户所购商品会丢失,无法满足个性化的需求,直接影响到站点的吸引力和影响力。作为一个电子商务网站,非常希望为用户提供个性化的功能,但由于技术难度、时间进度和开发成本等多方面因素的影响,个性化的构想仍难以实现。

ASP.NET 2.0技术提供了个性化服务解决方案的技术框架。该框架主要包括3项核心功能:个性化的用户配置、Web部件、成员和角色管理。提供个性化服务的Web站点必须在可控和安全的前提下运行。这就对用户信息的配置、存储和个性化服务方式等提出了要求。为显示个性化服务,Web站点主要执行3个步骤:识别用户身份、提供个性化服务体验和存储用户信息。

① 识别用户身份。Web站点只有了解用户登录身份才能依据身份提供相应功能服务。在这一步骤中首先要建立验证用户身份的机制,如在数据库中建立用户表,存储用户名和密码,当用户登录网站时,判断是匿名用户还是非匿名用户,如果是非匿名用户则获取用户输

入的用户名及密码和数据库中存储的用户名及密码进行对照，验证用户是否具有登录站点的合法身份。然后，站点还必须创建识别用户需求的机制。不同的用户具有不同的需求，匿名用户和登录用户所具有的功能有所不同，普通用户和管理员所具有的功能也不同。最后，Web 站点还必须创建管理用户的机制。

② 提供个性化服务的体验。Web 站点为匿名用户和登录用户提供的个性化服务是不同的，只有登录站点才能享受站点提供的众多的个性化服务。个性化的功能可以为客户提供自定义显示、持续跟踪用户操作信息等功能。所有的个性化功能都与用户身份关联。

③ 存储用户信息。必须能够存储用户的信息，不可能让客户每次登录站点都要重新注册和配置个人信息。能够存储客户个人信息和购物车信息 Web 站点提供的个性化功能紧密关联。

网上书店购物车的设计包含 4 个任务：设计待购图书详细信息显示界面、设计购物车内图书信息实体类和购物车类、设计匿名用户使用的购物车、实现匿名用户购物车数据向注册用户购物车的迁移。流程图如图 7-1 所示。待购图书信息显示界面提供每本图书的详细信息，以供客户选择，本模块中"购买"功能实现将图书信息添加到购物车中。购物车图书信息实体类，封装图书的编号、书名、销售价格、每本图书的购买数量和小计价格等信息。购物车类封装了图书信息实体集合，以及向购物车中添加图书、从购物车中删除图书、更新购物车中图书数量等操作。使用个性化配置实现匿名用户购物车，匿名用户选购的图书可以保存到匿名用户购物车中。实现注册用户购物车，当匿名用户注册登录网站时，实现匿名用户购物车内商品信息向注册用户购物内迁移。

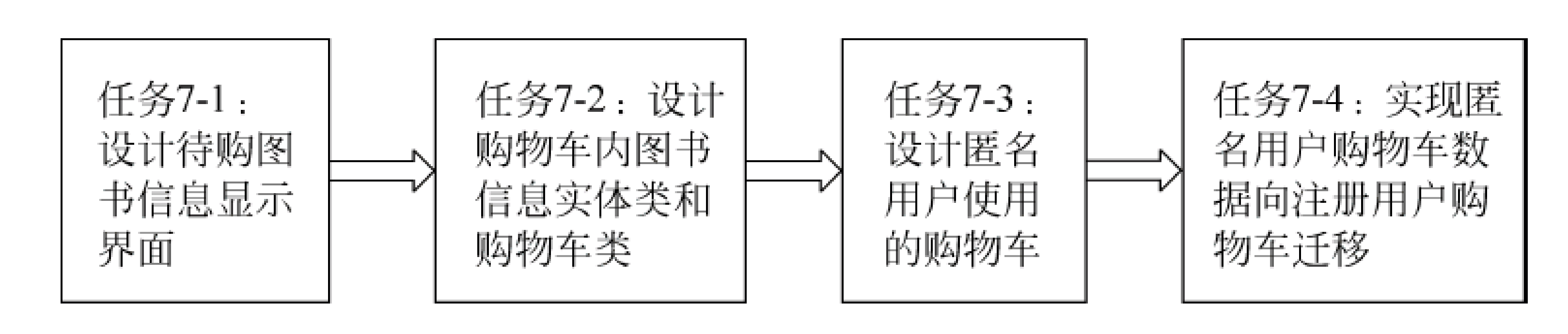


图 7-1 购物车管理模块流程图

7.3 相关知识

1. 个性化配置知识

在实现购物车时可以使用个性化的配置，在站点的 web.config 文件中添加 Profile 配置节，将购物车对象标识配置到配置节中，可以实现购物车个性化的访问。ASP.NET 2.0 提供的个性化用户配置功能可以实现将配置信息与单个用户关联，并采用持久化方式存储信息。配置信息可以是任何与用户有关的信息，如用户使用的主题等。所存储的配置信息可以是任何数据类型对象，例如，简单数据类型 String、Int 等，也可以是复杂自定义对象（如购物车对象）。单个用户可以是匿名用户也可以是注册用户，默认情况下，个性化用户配置功能只存储注册用户配置信息，但可以修改配置节相关内容允许保存匿名用户配置信息。默认情况下，持久化存储采用 SQLServer 数据库连接方式，并且无须自行创建和维护该数据库，这些工作由 ASP.NET 2.0 自行完成。

使用个性化用户配置功能主要包括两个步骤：一是配置应用程序以便启用和定义存储

用户跟踪用户信息的配置信息，这些工作在 web.config 文件的<profile>配置节中完成，例如，定义配置信息名称、数据类型，设置是否允许匿名用户存储有关信息等；二是使用与用户配置功能有关的强类型 API 实现对用户配置信息的存储、访问和管理等。

2. 哈希表(HashTable)

在.NET Framework 中，HashTable 是 System.Collections 命名空间提供的一个容器，用于处理和表现类似 key/value 的键值对，其中 key 通常可用来快速查找，同时 key 区分大小写；value 用于存储对应于 key 的值。HashTable 中 key/value 键值对均为 object 类型，所以 HashTable 可以支持任何类型的 key/value 键值对。HashTable 类的 Value 属性可以返回 ICollection 类型的数据集合，购物车存储的商品数据可以保存在 Value 属性中，ICollection 类型的数据集合可以轻松与 GridView 控件绑定，并将数据显示出来。

哈希表的简单操作如下：

在哈希表中添加一个 key/value 键值对：

```
HashtableObject.Add(key,value);
```

在哈希表中去除某个 key/value 键值对：

```
HashtableObject.Remove(key);
```

从哈希表中移除所有元素：

```
HashtableObject.Clear();
```

判断哈希表是否包含特定键 key：

```
HashtableObject.Contains(key);
```

3. 实体类

分层结构的数据库软件开发中，一般都要用到实体类。要了解实体类的作用，首先有必要了解对象关系映射 ORM(Object/Relation Mapping)思想。它是用来解决面向对象程序与关系型数据库之间一种对象关系模型的方案，通过在程序中定义对象模型，来映射数据库中的业务表。ORM 的好处显而易见，将数据库业务表映射为程序对象，对表字段的读写可通过对象相关属性操作，对表的查询可通过对象集合获取查询结果信息，对表的添加、删除、更新等操作同样可通过对象的相关方法来实现。在分层结构中，用户甚至不需关心数据库链接、数据管理等操作，直接使用相关操作接口，通过业务表映射的程序对象即可实现对数据的查询和管理操作。大多数情况下，程序中定义的对象模型，就是表的“实体映射”，即一张业务表映射到一个对象类，具体讲就是一个表名对应一个类名，表中各字段对应写成此类的各属性。这是 ORM 映射方案的基础。

项目 7 中使用实体类集合存放图书信息。然后设计一个购物车类对实体类进行操作，购物车类封装的是购物方法的实现。购物车内容的呈现使用 GridView 控件。用户购物界面使用 Datalist 控件设计，Datalist 控件与 SqlDataSource 数据源控件绑定，呈现数据库中图书的信息。

7.4 项 目 实 施

在接下来的内容里,将介绍子任务的实现过程。项目实施过程分解为如下的4个任务,每个任务都有若干步骤才能完成。下面具体介绍。

7.4.1 任务7-1 设计图书购买界面

1. 任务介绍

添加图书到购物车需要设计一个待购图书信息显示界面,显示每本图书的书名、市场价、会员价、订购数量、图书内容介绍等详细信息。本任务将介绍如何使用DataList控件结合SqlDataSource数据源控件设计待购图书的购买界面。

2. 任务分析

在图书购买界面中应包含一个描述订购动作的按钮,单击该按钮可实现将图书添加到购物车的过程,完成图书选购。使用SqlDataSource数据源控件从数据库中提取图书详细信息。使用DataList控件设计图书购买界面。将DataList控件和数据源控件绑定显示图书信息。

(1) 配置数据源控件SqlDataSource

在解决方案的Web文件夹中新建一个bookView.aspx文件,打开该文件的设计界面。从工具箱的"数据"选项卡中选择"SqlDataSource"数据源控件,拖放到设计界面中。

使用项目4中介绍的配置数据源知识为数据源配置Select语句。打开"配置Select语句"对话框,单击"指定来自于表或视图的列"选项,选择视图文件"View_Book",在需要呈现的字段前打勾,如图7-2所示。

设置Where子句的条件[bookid] = @bookid,参数@bookid绑定Label控件"labBookID"的Text属性(存储图书的编号)。

单击"下一步"按钮,进入"测试查询"对话框,单击"测试查询"可以对SQL语句进行测试,单击"完成"按钮,完成配置数据源的操作。

后台的BookView.aspx.cs文件中"Page_Load"方法的代码如下:

```
protected void Page_Load(object sender, EventArgs e) {
    if (!IsPostBack) {
        labBookID.Text = Request.QueryString["BookID"];
    }
}
```

从前页获取图书的编号存储到Label控件中,数据源从数据库查询图书信息参数用该Label控件获取。

(2) 使用DataList控件设计图书购买界面

从工具箱的"数据"选项卡选择DataList添加到设计界面中。在"DataList任务"对话

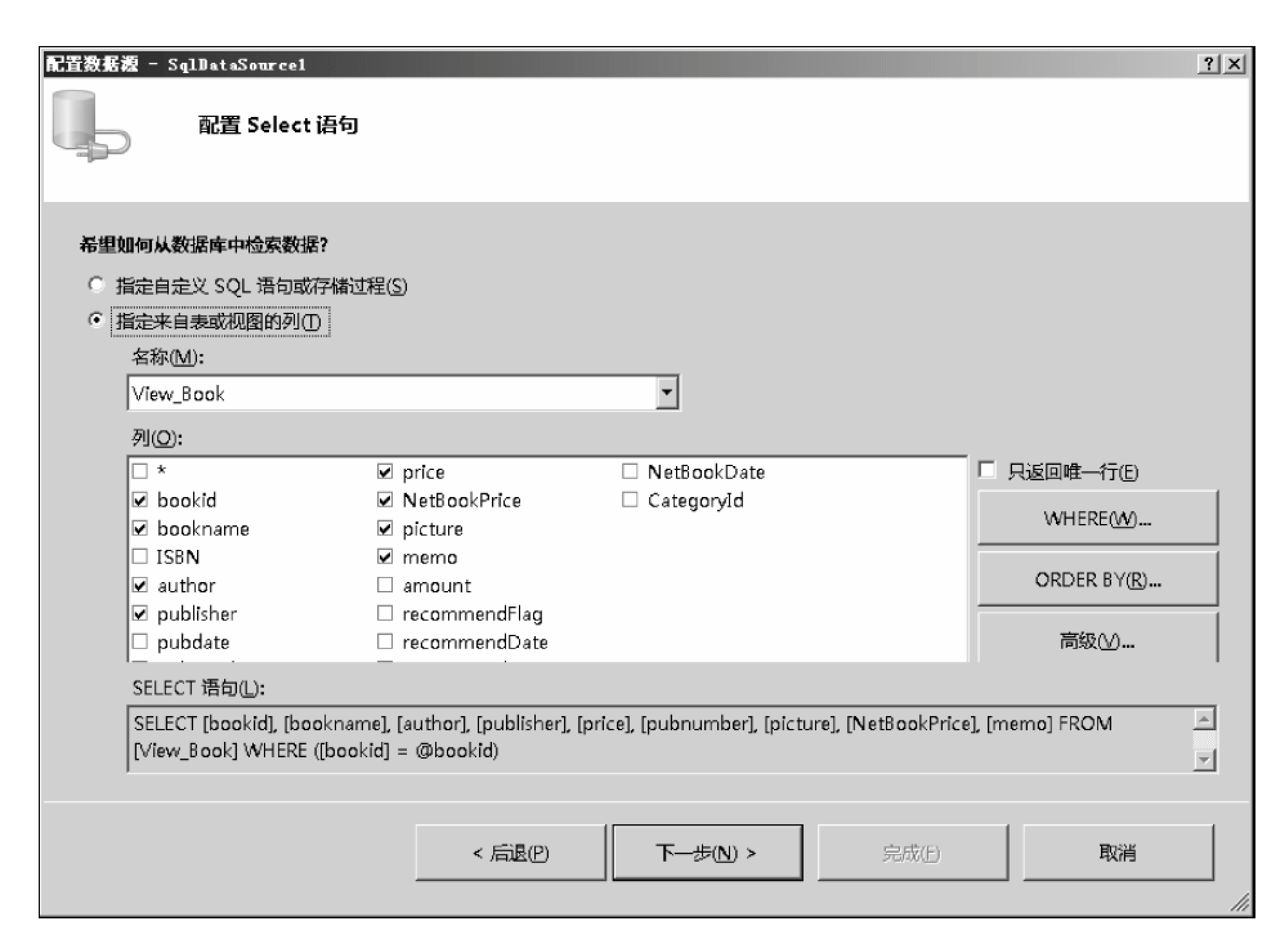


图 7-2　配置 Select 语句

框中选择步骤(1)中配置的"SqlDataSource1"数据源,将 DataList1 控件与数据源绑定,单击"刷新架构",结果如图 7-3 所示。

选择"DataList 任务"对话框中的"编辑模板",打开编辑模板对话框,如图 7-4 所示。DataList1 项模板中产生 9 个 Label 控件分别绑定 SqlDataSource1 的 9 个字段。

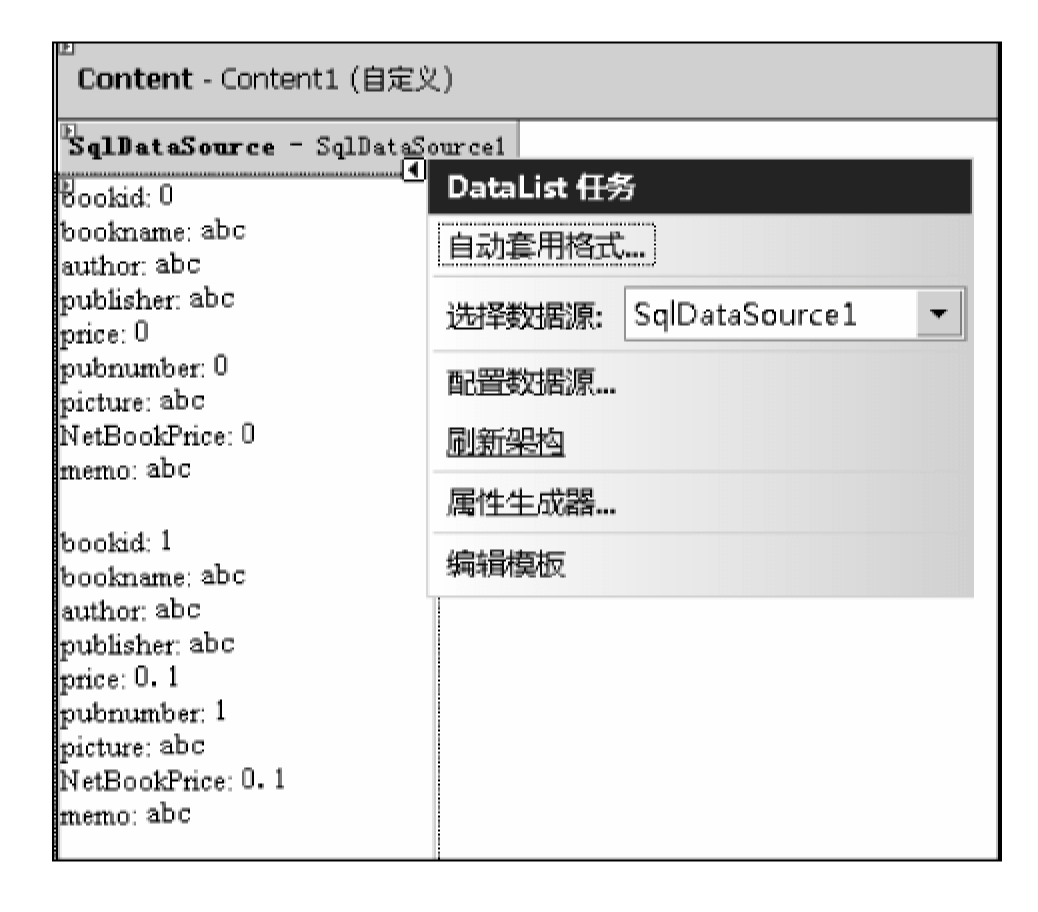


图 7-3　DataList1 控件绑定 SqlDataSource1 数据源

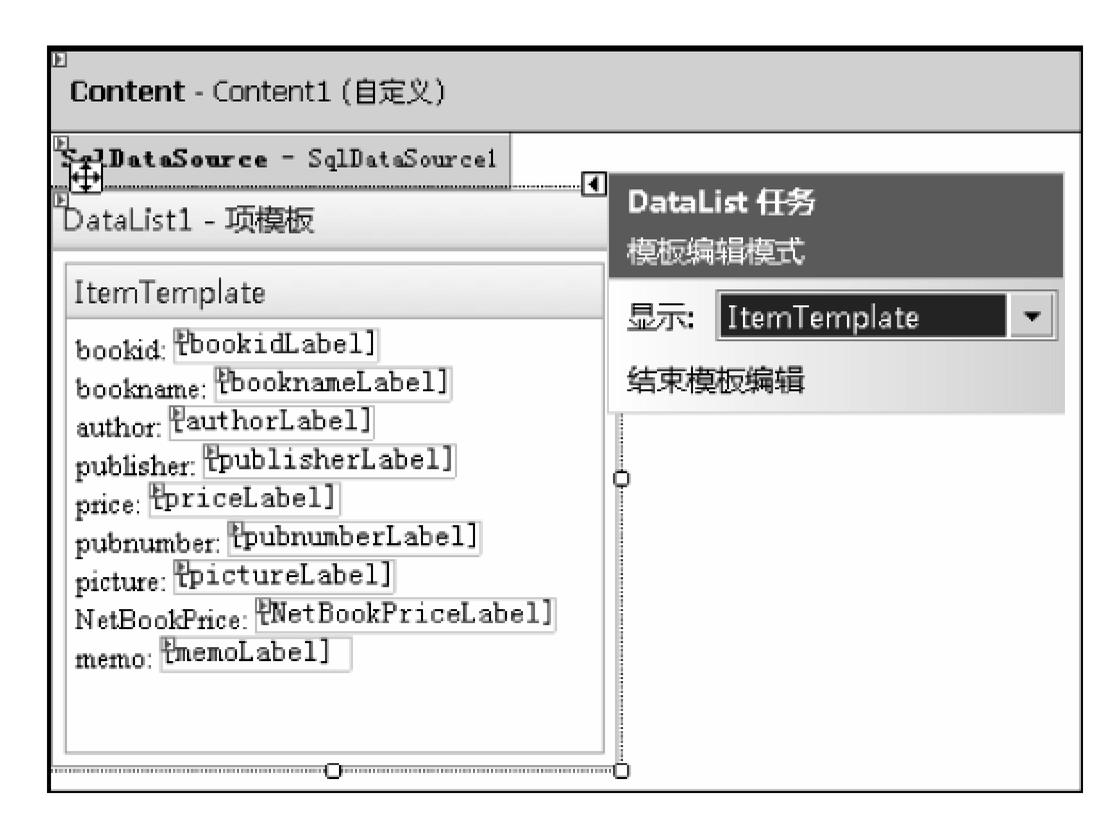


图 7-4　DataList1 模板编辑

对 Label 控件重新使用表格布局,布局界面如图 7-5 所示。其中 pictureLabel 标签控件绑定了图片的路径,更改为用 Image 控件描述。图中左上角打叉的控件。

SqlDataSource - SqlDataSource1
DataList1 - 项模板
ItemTemplate
[labTitle]
市场价: [priceLabel] [Label2] [Label3]
普通会员价: [NetBookPriceLabel]
3星级会员: [labPrice3]
5星级会员 [labPrice5]
购买
基本信息
【书名】 [booknameLabel]
【作者】 [authorLabel]
【出版社】 [publisherLabel]
【版本】 [Label1]
内容简介
[memoLabel]

图 7-5 重新布局过的 DataList1 项模板

页面代码描述如下：

```
<asp:Image ID = "Image1" runat = "server" Height = "200px" Width = "160px" ImageUrl = '<% #
Eval("picture") %>' />
```

ImageUrl 属性绑定数据表 t_book 的 picture 字段，可以从数据库中获取图片的位置，在生成页面时，该位置显示图片。

描述图书的市场价和网店销售价时，使用的页面代码如下：

```
<asp:Label ID = "priceLabel" runat = "server" Text = '<% # Eval ( "price", "{0:C}") %>'
ForeColor = "Red">
<asp:Label ID = "NetBookPriceLabel" runat = "server" Text = '<% # Eval("salePrice", "{0:C}")
%>' ForeColor = "Red">
```

在绑定市场价 price 和销售价 salePrice 时，使用格式控制符“{0:C}”，在页面呈现时，将自动以货币形式呈现。

添加按钮“LinkButton”，按钮的 ID 属性设为“lbtPurchase”，文本 Text 属性设为“购买”，将“commandargument”属性绑定到字段“bookid”，页面代码描述如下：

```
<asp:LinkButton ID = "lbtPurchase" runat = "server" commandargument = '<% # Eval("bookid")
%>' CommandName = "Purchase" OnCommand = "lbtPurchase_Command">购买</asp:LinkButton>
```

控件“lbtPurchase”的 Command 事件函数结构如下：

```
protected void lbtPurchase_Command(object sender, CommandEventArgs e){
    //进行添加图书到购物车的处理，将选中的图书信息加入到购物车。
    //后台代码在实现购物车操作时分析。
}
```

当单击按钮时，将执行添加图书到购物车的操作。

待购图书信息显示界面运行效果如图 7-6 所示。



图 7-6 图书购买界面

3. 任务完成总结

购书界面设计的关键点是将 DataList 控件与数据源控件 SqlDataSource 绑定，使用 DataList 控件的项模板重新构造界面。在 DataList 控件的项模板中使用 Image 控件显示图片，图片的路径参数通过 DataList 控件和数据源控件的绑定获取。

4. 课堂训练与知识拓展

在图书订购页面中，除了显示当前待购图书的信息外，一般在设计时，还会考虑在界面的左侧栏或右侧栏显示与购买图书相关的其他图书简要信息，以供客户选择。在待购图书信息显示界面的左侧栏使用 DataList 控件进行布局设计，显示与待购图书同类的图书，和最新出版的 10 本图书的简要信息。

7.4.2 任务 7-2 设计购物车商品信息类和购物车类

1. 任务介绍

购物车的每个商品可以用一个实体类进行描述。用户使用购物车需要能够将选购商品放置到购物车中，能够从购物车中移除商品，还要能够修改购物车中的商品数量，这些操作可以封装在一个购物车操作类中。本任务将介绍存储图书信息的实体类和购物车类的设计过程。

2. 任务分析

购物车中图书信息包含图书的编号、书名、价格、数量、小计等信息。小计是通过价格和

数量相乘的结果。购物车的每本图书可以用一个实体类(CartItem)描述。CartItem 类以面向对象的方式构建了购物车中的单条商品对象。购物车中一般有多本图书,可以用哈希表来存储,使用图书的编号作为哈希表的键,存储图书信息的实体对象作为哈希表的值。购物车类(ShoppingCart)对购物车的操作实际上是对哈希表的操作。购物车要能实现图书信息的添加、从购物车中移除图书信息和重新设置购物车中图书数量的功能。购物车商品信息类图和购物车类图如图 7-7 所示。



(a) 购物车商品信息类图



(b) 购物车类图

图 7-7 购物车商品信息类图和购物车类图

(1) 设计购物车商品信息类

CartItem 类由 3 部分构成,私有成员变量(字段)、属性和构造函数。外部的图书编号、书名、定价参数值在类进行实例化时由构造函数传给内部私有成员变量,购买数量默认值为 1,由私有成员变量 m_Quantity 初始化。描述图书编号、书名、图书价格和单件商品小计价格的属性设计成只读属性,而购物车中商品的数量客户需要更新,设计成可读写的属性。单件商品小计价格是由内部成员变量 m_BookPrice 和 m_Quantity 相乘后的结果。

在解决方案的 App_Code 文件夹下添加 CS 文件 CartItem. cs,设计 CartItem 类代码如下:

```
[Serializable]
public class CartItem{
    private int m_BookID;
    private string m_BookName;
    private decimal m_BookPrice;
    private int m_Quantity = 1;


    //定义图书编号属性
    public int BookID{
        get { return m_BookID; }
    }

    //定义图书的书名属性
    public string BookName {
        get { return m_BookName; }
    }

    ///定义图书的价格属性
    public decimal BookPrice {
        get { return m_BookPrice; }
    }

    //定义订购数量属性
```

```
        public int Quantity
        {
            get { return m_Quantity ; }
            set { m_Quantity = value; }
        }

        //定义单件商品小计价格属性
        public decimal SubTotalPrice {
            get { return m_BookPrice * m_Quantity; }
        }

        //图书业务实体类构造函数,参数图书编号、书名、购买价格
        public CartItem(int bookID,string bookName,decimal bookPrice) {
            m_BookID = bookID;
            m_BookName = bookName;
            m_BookPrice = bookPrice;
        }
    }
```

（2）设计购物车类

购物车类 ShoppingCart 主要实现购物车内图书信息数据的添加、删除和修改的功能。字段 m_HashCart 为哈希类型的字段，以哈希表方式存储购物车内所有商品数据。ShoppingCart 类还包含两个属性。CartItems 属性返回购物车内所有商品数据，属性类型为 ICollection 类型，该类型包含在 System. Collections 命名空间中，因此，设计 ShoppingCart 类需要引用命名空间“using System. Collections”。ShoppingCart 类的另一个属性 TotalPrice 用于获取购物车中商品的总价，具体设计可以参考下面的代码设计。此外，ShoppingCart 类还包含 3 个方法，AddBook 方法实现向购物车中添加图书，RemoveBook 方法实现从购物车中删除图书，SetQuantity 方法实现更改购物车中商品购买数量。这 3 个方法分别使用了 HashTable 类提供的 Add 和 Remove 方法，以及 HashTable 类的索引机制。每本图书的“BookID”作为哈希表的键值，每个键值所对应的值为相应的图书实体对象“CartItem”。

在解决方案的 App_Code 文件夹下添加 CS 文件 ShoppingCart. cs，设计 ShoppingCart 类代码如下：

```
    [Serializable]
    public class ShoppingCart{
        //定义购物车对象,Hashtable 类型对象
        private Hashtable m_HashCart = new Hashtable();

        // 定义购物车的商品订单属性
        public ICollection CartItems{
            get { return m_HashCart.Values; }
        }

        //定义购物车中商品总价属性
        public decimal TotalPrice {
            get {
```

```
            decimal sum = 0;
            foreach (CartItem item in m_HashCart.Values) {
                sum + = item.BookPrice * item.Quantity;
            }
            return sum;
        }
    }

    // 定义向购物车中添加图书的成员函数,参数图书编号、书名和购买价格
    public void AddBook(int bookID, string bookName, decimal bookPrice) {
        //依据图书编号到购物车中查询图书对象
        CartItem item = (CartItem)m_HashCart[bookID];

        //如果购物车中没有该图书,添加到购物车中,否则订购数量加 1
        if (item== null) {
          m_HashCart.Add(bookID, new CartItem(bookID, bookName, bookPrice));
        }
        else {
            item.Quantity++;
            m_HashCart[bookID] = item;
        }
    }

    //定义从购物车中移除图书的成员函数,参数图书编号
    public void RemoveBook(int bookID) {
        CartItem item = (CartItem)m_HashCart[bookID];
        if (item != null) {
              m_HashCart.Remove(bookID);
        }
    }

    //定义设置订购数量的成员函数,参数图书编号和订购数量
    public void SetQuantity(int bookID, int quantity) {
        CartItem item = (CartItem)m_HashCart[bookID];
        if (item != null && quantity > 0) {
             item.Quantity = quantity;
             m_HashCart[bookID] = item;
        }
    }
}
```

3. 任务完成总结

本任务完成了购物车商品信息类和购物车类设计。CartItem 类和 ShoppingCart 类构建的哈希表将被以二进制序列化方式存储到配置数据库的 aspnet_Profile 表中,因此,两个类的类名前都用“[ ]”标记一个属性 Serializable。该属性指示类被序列化。序列化是指将对象实例的状态存储到存储介质的过程。在此过程中,先将对象公共字段、私有字段以及类名转换为字节流,然后再把字节流写入数据流。对对象进行反序列化时,将创建出与原对象完全相同的副本。序列化和反序列化的过程是由 ASP.NET 2.0 自动完成的。

4. 课堂训练与知识拓展

客户购买图书金额达到一定的级别后，将获得网站的折扣优惠，网上书店往往根据客户的级别，给客户一个优惠价，在进行购物车设计时，购物车内单件商品要显示两种价格，图书的市场价和会员价，会员价是网站依据客户的级别给客户的优惠价。购物车要计算两种价格的小计及总差价。依据上述的描述改写购物车商品信息类和购物车类，实现会员价的计算。

7.4.3　任务 7-3　实现匿名用户使用购物车

1. 任务介绍

在传统的电子商务网站的设计中，客户访问电子商务网站时，一般先浏览页面，查看自己感兴趣的商品，当客户要使用购物车时，需要登录，这会限制匿名用户的购物热情，将失去一部分潜在的客户。如果匿名用户没有在网站注册，强制用户注册，会打断用户的购物流程。即使已经注册的用户可能只是浏览商品，也未必会购买商品。这些问题在 ASP.NET 2.0 框架下可以通过个性化的配置，设置匿名购物车解决。

2. 任务分析

购物车的界面可以用 GridView 控件实现。购物车类的 CartItems 属性保存购买的商品信息，为 GridView 控件提供数据源。使用个性化服务配置购物车。匿名用户登录网站时，网站分配匿名用户 ID，匿名用户可以使用购物车的信息保存到配置数据库中。当匿名用户再一次登录网站时，可以从配置文件中获得购物车信息。

下面的步骤将介绍如何设计和使用匿名购物车。

(1) 设计购物车界面

购物车界面包含两方面的内容：一是购物车内图书的基本信息，如图书的编号、书名和价钱等；二是对这些基本信息的操作，如从购物车中删除图书、更新单件图书的购买数量、购买产生订单等。界面如图 7-8 所示。

购物车

清除	图书编号	图书名称	销售价格	价格小计	更新数量
清除	31	西游记	￥19.00	￥19.00	1
清除	30	深入理解.NET	￥95.00	￥95.00	1
清除	29	写给大家看的CSS书	￥49.00	￥49.00	1
清除	1	明朝那些事儿	￥23.00	￥46.00	2
					更新 购买

总价：￥209.00

图 7-8　购物车界面信息

设计购物车界面的代码如下：

```
<asp:GridView ID = "gvCart"
            runat = "server"
            DataKeyNames = "BookID"
            OnSelectedIndexChanged = "gvCart_SelectedIndexChanged" ShowFooter = "True">
```

```
            <Columns>
                <asp:CommandField HeaderText = "清除"
                                  SelectText = "清除"
                                  ShowSelectButton = "True" >
                    <ItemStyle ForeColor = "Blue" />
                </asp:CommandField>
                <asp:BoundField DataField = "BookID"
                                HeaderText = "图书编号" />
                <asp:BoundField DataField = "BookName"
                                HeaderText = "图书名称" />
                <asp:BoundField DataField = "BookPrice"
                                DataFormatString = "{0:c}"
                                HeaderText = "销售价格" />
                <asp:BoundField DataField = "SubTotalPrice"
                                DataFormatString = "{0:c}"
                                HeaderText = "价格小计" />
                <asp:TemplateField HeaderText = "更新数量">
                    <ItemTemplate>
                        <asp:TextBox ID = "txbBookQuantity"
                                     runat = "server"
                              OnTextChanged = "txbBookQuantity_TextChanged"
                                      Text = '<% # Bind("Quantity") %>'
                                      Width = "50px"/>
                    </ItemTemplate>
                    <FooterTemplate>
                        <asp:LinkButton ID = "lbtUpdate"
                                        runat = "server"
                                   ForeColor = "Blue">更新</asp:LinkButton>
                        <asp:LinkButton ID = "lbtBuy"
                                        runat = "server"
                                OnClick = "lbtBuy_Click">购买
    </asp:LinkButton>
                    </FooterTemplate>
                </asp:TemplateField>
            </Columns>
            <EmptyDataTemplate>
                购物车没有图书信息!
            </EmptyDataTemplate>
        </asp:GridView>
    <asp:Label ID = "labTotalPrice" runat = "server" />
```

控件 GridView 的属性 DataKeyNames 的值为图书编号 BookID,通过 BookID 可以唯一标识一本图书。属性 OnSelectedIndexChanged 定义事件 gvCart_SelectedIndexChanged 当选择一本图书时执行该事件,购物车设计中将该事件定义为删除图书的操作,如 GridView 控件第 1 列的配置。事件代码将在后面的步骤分析。第 2~5 列分别绑定图书的编号、书名、销售价格和小计价格。DataField 属性分别绑定购物中图书业务实体类 CartItem 的属性。HeaderText 属性值为购物车标题栏显示的文字。DataFormatString＝"{0:c}"定义了货币数值的显示格式。购物车最后一列购买数量可以更新,使用模板列设计。模板项 ItemTemplate 中包含一个 TextBox 控件,让用户填写数量。TextBox 控件的属性 OnTextChanged 定义事件 txbBookQuantity_TextChanged,当购买数量发生更新时要

更新购物车中相应图书的数量，事件代码将在下面的步骤中介绍。

最后一列的页脚模板项 FooterTemplate 中设置了两个 LinkButton 按钮用于执行数量更新和购买图书并产生订单。显示页脚需要将 GridView 控件的属性 ShowFooter 设置为 True。模板项 EmptyDataTemplate 设置当数据库源为空时的提示信息。

购物车中图书的总价格用 Label 控件的 labTotalPrice 属性呈现。

(2) 匿名用户购物车个性化设置

匿名用户登录网站，网站将给匿名用户产生一个 ID，用于标识匿名用户的身份。使用 aspnet_regsql 工具在 NetBook 数据库中创建存储 profile 信息的表结构，存储过程及视图。匿名用户登录网站的信息将写入表 aspnet_Users 中。匿名用户使用购物车的信息将写入表 aspnet_Profile 中。

设计匿名用户购物车需要对 Web. Config 进行设置。

① 配置个性化服务默认数据库连接词。在<configuration>配置节的子配置节<connectionStrings>配置节下添加访问 NetBook 数据库的连接词。配置代码段如下：

```
<connectionStrings>
    <remove name="LocalSqlServer" />
    <add name=" LocalSqlServer "
        connectionString="Data Source=.;Initial Catalog=NetBook;User ID=sa;Password=
sa@sqlserver"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
```

首先移除系统默认的数据库连接词“LocalSqlServer”，然后定义 NetBook 数据库连接词“LocalSqlServer”，用户的登录 ID 和密码依据实际情况设置。

<system. web>配置节的子配置节<profile>的属性 defaultProvider 设置默认连接对象名 MyProfileProvider，在<profile>子配置节<providers>下添加连接配置项。连接配置项的 connectionStringName 属性值与上面介绍的<connectionStrings>配置节访问数据库的连接词要一致。配置代码段如下：

```
<system.web>
  <profile defaultProvider="MyProfileProvider">
    <providers>
      <add name="MyProfileProvider"
          type="System.Web.Profile.SqlProfileProvider"
          connectionStringName=" LocalSqlServer " />
    </providers>
  </profile>
</system.web>
```

② 设置匿名用户访问的配置项。允许匿名用户访问数据库可以通过设置 Web. Config 的配置节来达到目的。在<system. web>配置节下添加允许匿名用户访问的配置项<anonymousIdentification>，Enabled 属性的值设为 True。配置代码段如下：

```
<system.web>
  <anonymousIdentification enabled="true"/>
</system.web>
```

③ 设置个性化的匿名购物车配置项。允许匿名用户使用购物车，需要在<profile>配置节下增加<properties>配置节，在<properties>配置节下添加 ShoppingCart 配置项，type 属性指出项目类型为自定义的购物车类，serializeAs="Binary"指出购物车将被序列化为二进制数据库存储到数据库中，设置 allowAnonymous="true"允许匿名用户使用购物车。配置代码段如下：

```
<profile defaultProvider = "MyProfileProvider">
      <properties>
           <add name = "ShoppingCart"
                type = "ShoppingCart"
                serializeAs = "Binary"
                allowAnonymous = "true"/>
        </properties>
</profile>
```

(3) 实现图书信息添加到购物车中功能

本步骤将实现任务 7-1 中的图书信息加入购物车的事件代码。从 profile 配置信息中获取购物车，当匿名用户第一次访问网站时，配置信息中的购物车对象为空，需要创建购物车对象，并保存到配置项中。从配置项中获取购物车类对象，调用购物车类对象的 AddBook 方法将图书信息保存到购物车对象中。

添加图书信息到购物车的代码如下：

```
protected void lbtPurchase_Command(object sender, CommandEventArgs e){
  if (e.CommandName== "Purchase") {
     //从图书购买提交事件的 CommandArgument 属性中的获取图书的编号，
     //并转换为整型
     int bookID = int.Parse(e.CommandArgument.ToString());

     //获取 DataList 控件的模板项
     DataListItem dlItem = (sender as LinkButton).Parent as DataListItem;

     //获取保存书名和书价的 Label 控件
     Label labBookName = dlItem.FindControl("booknameLabel") as Label;
     Label labBookPrice = dlItem.FindControl("priceLabel") as Label;

    if (labBookName!= null && labBookPrice!= null) {
        string bookName = labBookName.Text;

        //货币类型的图书价格转换为数值类型
        decimal bookPrice = decimal.Parse(labBookPrice.Text, System.Globalization.
NumberStyles.Currency);

       //如果 Profile 对象中存储的购物车对象为 null，
       //则创建一个购物车对象，保存到 Profile 对象中
       if (Profile.ShoppingCart== null) {
             Profile.ShoppingCart = new ShoppingCart();
       }

       //将购买的图书添加到 Profile 对象的购物车中
```

```
            Profile.ShoppingCart.AddBook(bookID, bookName, bookPrice);
        }
      }
    }
```

（4）实现呈现购物车中的图书信息功能

从 profile 配置信息中读取购物车对象 ShoppingCart，该对象的属性 CartItems 保存了购物车中的商品信息，将 CartItems 作为数据源到绑定 GridView 类型的控件 gvCart 的 DataSource 属性上，执行 gvCart 对象的 DataBind()方法可以呈现购物车的图书信息。使用 ShoppingCart 对象的 TotalPrice 属性获取购物车中所有图书的总价格。

呈现购物车中图书信息的代码如下：

```
protected void BindShoppingCart(){
    if (Profile.ShoppingCart!= null) {
        //将 Profile 对象中的购物车商品信息绑定到 GridView 类型的显示控件
        gvCart.DataSource = Profile.ShoppingCart.CartItems;
        gvCart.DataBind();

      //计算购物车中图书的总价格
      if (gvCart.FooterRow!= null){
       labTotalPrice.Text = "总价：" + Profile.ShoppingCart.TotalPrice.ToString("c");
      }
    }
}
```

（5）实现删除购物车中的图书信息功能

首先从 gvCart 对象的 SelectedDataKey 属性中获取被选中行的 DataKey 对象，DataKey 对象的 Value 属性保存了图书的编号 BookID。将图书编号作为购物车对象 ShoppingCart 的方法 RemoveBook 的参数，实现从购物车中删除图书的操作。

删除购物车中图书信息的代码如下：

```
protected void gvCart_SelectedIndexChanged(object sender, EventArgs e){
     int bookID = (int) TotalPrice gvCart. gvCart SelectedDataKey.Value;
     Profile.ShoppingCart.RemoveBook(bookID);
     gvCart.SelectedIndex = -1;
     BindShoppingCart();
}
```

（6）实现更新购物车中图书的购买数量功能

读取 profile 配置信息中购物车对象 ShoppingCart，使用 ShoppingCart 对象的方法 SetQuantity 更新购物车中图书的订购数量，执行 SetQuantity 方法时，传输两个参数“图书编号”和“订购数量”。更新成功则重新呈现更新后的结果，更新失败则给出“购买数量为大于 0 的整型数字！”的提示信息。

更新购物车中图书购买数量的代码如下：

```
protected void txbBookQuantity_TextChanged(object sender, EventArgs e){
     bool isSucceed = false;
```

```
        //获取 GridView 控件中包含 TextBox 对象的行对象
        TextBox txbQuantity = sender as TextBox;
        GridViewRow row = txbQuantity.Parent.Parent as GridViewRow;

        if (!string.IsNullOrEmpty(txbQuantity.Text)) {
            try {
              //获取订购数量
              int intQuantity = int.Parse(txbQuantity.Text);
              if (intQuantity > 0) {
                  //获取图书的编号
                  int bookID = int.Parse(gvCart.DataKeys[row.RowIndex]["BookID"].ToString());

                  //使用 ShoppingCart 对象的 SetQuantity 方法更新订购数量
                  Profile.ShoppingCart.SetQuantity(bookID, intQuantity);

                  //更新完毕,重新呈现购物车更新后的结果
                  BindShoppingCart();
                  isSucceed = true;
              }
              else {
                  isSucceed = false;
              }
            }
            catch {
                isSucceed = false;
            }
        }

        if (!isSucceed) {
            txbQuantity.ForeColor = System.Drawing.Color.Red;
            labMsg.Text = "购买数量为大于 0 的整型数字!";
        }
        else {
            txbQuantity.ForeColor = System.Drawing.Color.Black;
            labMsg.Text = string.Empty;
        }
    }
```

3. 任务完成总结

本任务中使用个性化配置实现了匿名用户使用购物车。使用 GridView 设计购物车显示界面。从个性化配置对象 profile 中读取购物车对象,实现向购物车中添加图书信息、从购物车中删除图书信息、更新购物车中图书购买数量等购物车常用操作方法。

4. 课堂训练与知识拓展

结合网店的设计要求,将匿名用户使用购物车的功能模块添加到应用程序网站中,并实现清除购物车内商品信息的功能。

7.4.4　任务 7-4　实现匿名用户购物车到注册用户购物车的迁移

1. 任务介绍

任务 7-3 解决了匿名用户使用购物车功能。网站允许匿名用户登录，并使用购物车选商品，这时购物车中显示的商品是用户在匿名方式下所选择的商品列表，当用户下订单购买购物车中的商品时，需要用户使用注册账号登录站点，才可以继续进行购买操作。用户的购买操作是针对注册用户的购物车进行的，因此，需要将匿名用户购物车中的商品列表转移到注册用户购物车中，实现匿名用户向注册用户的迁移。

2. 任务分析

用户未登录站点时不能进行下订单的操作，用户单击图 7-8 购物车界面的"购买"按钮，应首先验证用户是否进行登录，如果用户没有登录则转向登录界面。代码描述如下：

```
if (!User.Identity.IsAuthenticated){
    Response.Redirect("~/Web/Logon.aspx");
}
```

实现用户验证需要在 web.config 文件对验证方式和默认登录页面进行配置。用户登录需要一个账号，可以使用项目 6 中介绍的创建注册用户功能页面建立注册用户，也可以利用网站管理工具创建用户账号。当用户使用分配的账号登录网站时，系统需要实现匿名用户向注册用户的迁移，将匿名用户购物车内商品列表迁移到注册用户购物车内。

当完成数据迁移后，用户退出登录，关闭客户端浏览器，然后再打开浏览器以匿名用户身份访问网站，匿名用户购物车中商品列表将被清空，应用程序为减少无用数据，应删除上次匿名用户所生成的数据。当用户使用上次登录站点的账号登录网站时，购物车中将显示用户购买的商品列表，表明匿名购物车中数据已转移到注册用户购物车中。

(1) 用户登录验证配置

实现用户登录和验证功能的第一步是对 Web.Config 文件进行配置，结合购物车的应用实例，配置的内容主要包括验证方式、设置默认登录页面、定义用户配置属性等。用户验证是基于表单的验证，只需在＜system.web＞配置节下添加以 Form 方式对用户的验证，默认登录页面为 Web 文件夹下的 Logon.aspx 页面。配置节代码如下：

```
<authentication mode = "Forms">
    <forms loginUrl = "~/Web/Logon.aspx"/>
</authentication>
```

(2) 利用网站管理工具创建用户账号

本步骤中利用网站管理工具创建一个用户，并为其设置密码。单击 Visual Studio 2005"网站"菜单中的"ASP.NET 配置"选项可以调用网站的配置工具，如图 7-9 所示。



图 7-9　调用 ASP.NET 配置工具

单击“ASP.NET 配置”选项，Visual Studio 2005 将启动一个内置的 WebService，并且在浏览器中加载网站管理工具。使用该工具可以对站点安全、应用程序配置和提供程序配置进行设置。单击“安全”选项卡进入如图 7-10 所示界面。



图 7-10　网站管理工具界面(“安全”选项卡)

该页面提供对网站的安全配置，其中包括安全配置向导、用户、角色和访问规则的各种配置项等。单击“创建用户”链接，可以进入创建用户的页面，如图 7-11 所示。



图 7-11　创建用户界面

添加新用户界面要求设置用户名、密码、电子邮件、安全提示问题和安全答案。本例中输入用户名"netbook",密码为"netbook@126.com",其他内容可以任意填写。单击"创建用户"按钮,账号创建成功。用户可以使用项目5中的登录功能登录站点。

(3) 实现匿名用户购物车配置数据向注册用户配置数据迁移

实现匿名用户购物车数据向注册用户购物车数据迁移的关键是实现 ProfileModule 类 MigrateAnonymous 事件。该事件在包含用户配置属性数据的匿名用户登录时发生,对应的事件处理程序是 Profile_MigrateAnonymous。与其他事件处理程序不同的是该事件处理程序必须在 Global.asax 文件中定义。Global.asax 文件应创建在网站的根目录下。



图 7-12 调用网站"添加新项"

单击 Visual Studio 2005"网站"菜单中的"添加新项"选项,如图 7-12 所示,可以打开"添加新项"对话框,如图 7-13 所示。选择"全局应用程序类",文件名默认为 Global.asax,选择语言"Visual C#",单击"添加"按钮将在应用程序根目录下创建全局应用程序类文件 Global.asax。

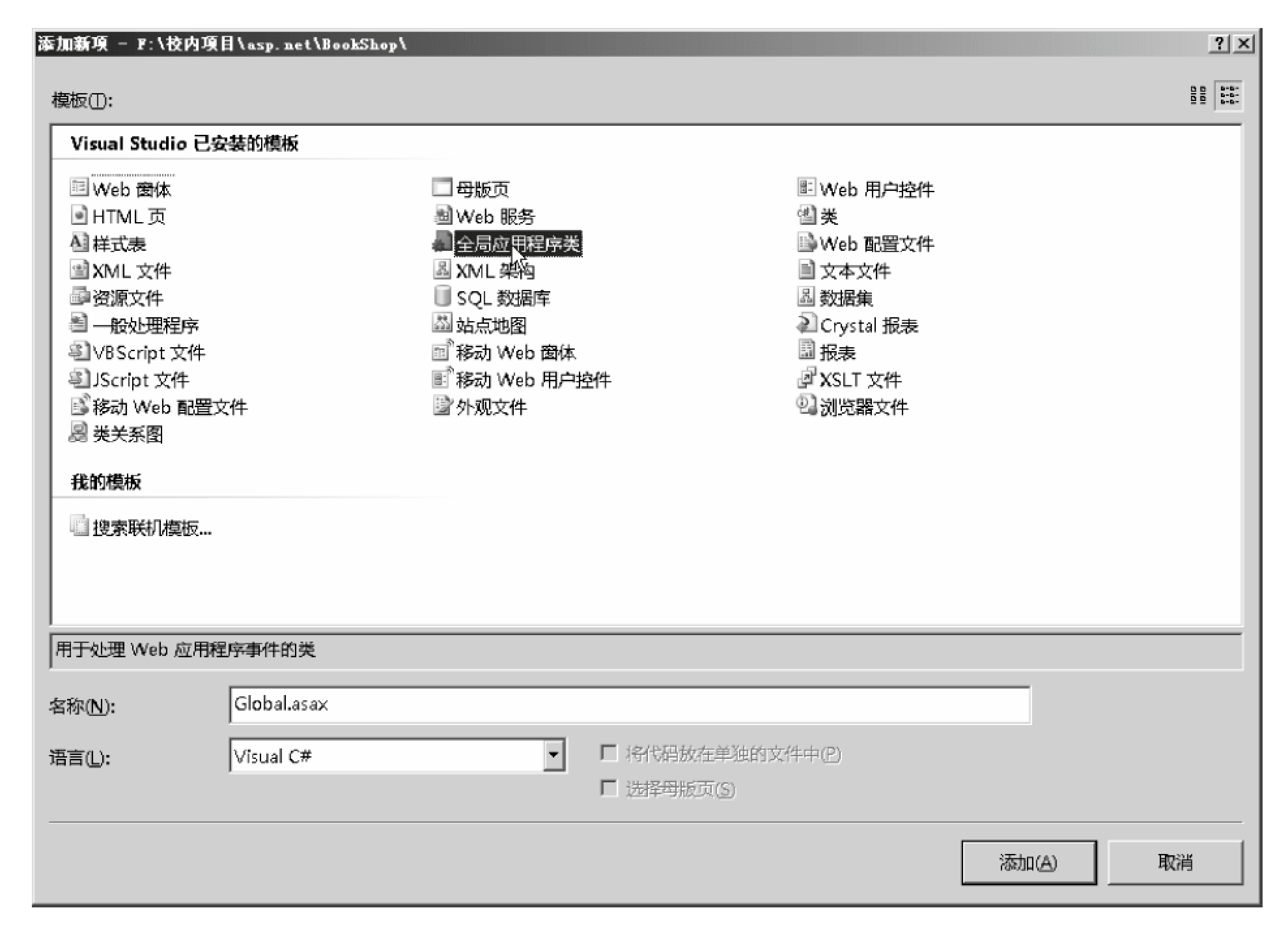


图 7-13 "添加新项"对话框

在 Global.asax 中添加事件处理程序 Profile_MigrateAnonymous,事件处理程序主要实现3个任务:一是从匿名用户的 Profile 属性中复制相关信息,并将其存储到注册用户的 Profile 属性中;二是为减少数据冗余删除数据库中存储的匿名用户的用户配置数据;三是

删除匿名用户标识，以免再次为登录的用户激发 MigrateAnonymous 事件。

事件处理程序的代码如下：

```
void Profile_MigrateAnonymous(object sender,ProfileMigrateEventArgs pe){
        //获取匿名用户的 Profile 属性
        ProfileCommon anonProfile = Profile.GetProfile(pe.AnonymousID);

        //如购物车中商品总价不为 0,则将匿名用户配置对象中购物车信息迁移到登录用户配置对象中
        if(anonProfile.ShoppingCart.TotalPrice!= 0) {
            Profile.ShoppingCart = anonProfile.ShoppingCart;
        }

        //从 aspnet_Users 表中删除匿名用户数据
        Membership.DeleteUser(pe.AnonymousID);

        //从 aspnet_Profile 表中删除匿名用户的 Profile 数据
        ProfileManager.DeleteProfile(pe.AnonymousID);

        //删除匿名用户标识
        AnonymousIdentificationModule.ClearAnonymousIdentifier();
}
```

在调用 ASP.NET 页面时，ASP.NET 创建了一个类 ProfileCommon，该类继承于 ProfileBase 类，用于强类型在 Web.Config 文件中定义的 profile 属性。从 ProfileMigrateEventArgs 事件委托中可以获得匿名用户的 ID (AnonymousID)。上述代码首先根据匿名用户 ID 获取其对应的 Profile 属性，该对象保存了用户在匿名身份下所存储的购物车商品数据，然后，将这些数据存储到注册用户的 Profile 属性中。

为了减少冗余数据，调用 Membership 类的 DeleteUser 方法从 aspnet_Users 表中删除匿名用户数据，调用 ProfileManager 类的 DeleteProfile 方法删除匿名用户存储在 aspnet_Profile 表中匿名用户配置属性数据。为了不让 MigrateAnonymous 事件再次激发，还需要调用 AnonymousIdentificationModule 类的 ClearAnonymousIdentifier 方法删除匿名用户标识。

3. 任务完成总结

任务 7-4 中讨论了匿名用户数据向注册用户的迁移，匿名用户登录为注册用户时引发 Global.asax 中的事件处理程序 Profile_MigrateAnonymous，在事件代码中完成匿名用户数据向注册用户数据迁移。为防止迁移重复发生，迁移成功后，应删除匿名用户数据信息。

4. 课堂训练与知识拓展

用户在登录网站后，将匿名用户购物车的图书列表迁移到注册用户购物车中，暂存在网站的数据库，下一次用户访问网站，继续匿名使用购物车购买图书。当用户再一次使用上次登录网站的账号登录网站时，匿名用户购物车中的图书列表将迁移到注册用户购物车中，覆盖注册用户购物车中原来图书的信息。用户上次存储在网站中购物车图书信息将丢失。请改写 Profile_MigrateAnonymous 事件处理程序的代码，实现匿名用户购物车图书列表和注

册用户购物车图书列表的合并。

7.5 项目总结

项目7通过4个任务完成网上购物车的设计，首先在任务7-1中介绍购物界面的设计。然后，任务7-2中介绍图书商品实体类的设计和购物车类的设计。在介绍购物车类设计时，按添加图书到购物车、显示购物车图书信息、从购物车中删除指定图书、更新购买数量的顺序进行介绍。任务7-3中详细介绍了匿名用户使用购物车的设计过程。最后，在任务7-4中实现了匿名用户购物车数据信息向注册用户购物车数据信息的迁移。

7.6 项目实训

1. 任务描述

会员客户在网上书店购书的总价达到一定的金额，网站会给会员一个级别代号。一般网站在设置员级别时有5级，每个级别享受的折扣不一样。根据任务描述，扩展项目7中的购物车，使得网上购物车可以根据会员客户级别自动计算商品的折扣价，并计算客户购物车商品小计价和总价。

2. 任务要求

① 使用项目4中介绍的数据库表设计方法，设计一个商品折扣表和一个关系表，折扣表中存储依据购书金额设定的折扣信息，关系表建立折扣与会员的关系，表中存储会员客户ID与折扣ID。

② 改写项目7的购物车中商品实体类，增加折扣价，折扣价是根据会员级别动态变化的。在显示购物车商品信息时，同时显示折扣率。

③ 匿名用户可以使用购物车，匿名用户向注册用户迁移时，如果注册用户购物车存在图书信息，则应实现两个购物车图书信息的合并。

网上书店客户订单管理

8.1 项 目 介 绍

订单管理是网上书店系统的重要功能。客户浏览网店,选择满意的商品,接下来需要付账购买。对于客户选择的商品要生成订单,网店工作人员查阅订单可以知道客户的商品应配送到哪里,客户查阅订单可以知道自己的商品是否已经配送。一个简化的订单管理功能应包括产生订单的购物流程、网店管理人员对订单处理和客户对订单跟踪与查询等子功能。项目 8 中将介绍客户订单管理功能的设计过程。

8.2 项 目 分 析

下面从系统应用架构和数据库表设计两个方面对项目做一简单分析。

1. 系统架构分析

在软件体系架构设计中,分层式结构是最常见,也是最重要的一种结构。微软推荐的分层式结构一般分为三层,从下至上分别为:数据访问层、业务逻辑层、表示层,如图 8-1 所示。

数据访问层有时也称持久层,其功能主要是负责数据库的访问。简单地说就是实现对数据表的 Select、Insert、Update、Delete 等操作。业务逻辑层是整个系统的核心,它与系统的业务(领域)有关,表示层的数据一般递交给逻辑层进行业务处理,如果涉及数据库的访问,则调用数据访问层。表示层是系统的 UI 部分,负责使用者与整个系统的交互。在这一层中,理想的状态是不应包括系统的业务逻辑。表示层中的逻辑代码仅与界面元素有关。

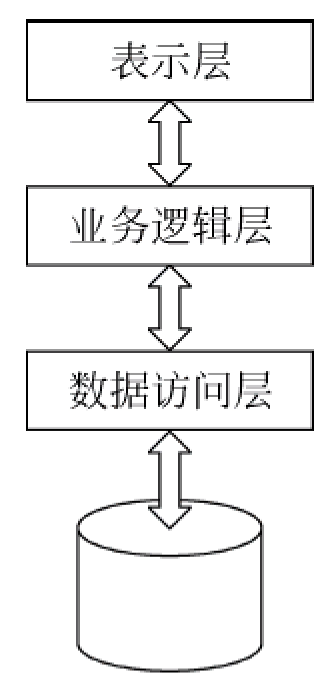


图 8-1 三层应用程序架构

本项目将使用三层应用程序架构技术完成订单管理功能的设计。设计过程分为 4 个任务。任务 8-1 使用用户控件设计图书订购流程界面,完成用户与系统的订购交互。任务 8-2 设计数据库操作重用类,构建访问数据库的公共执行逻辑。任务 8-3 设计订购流程业务类,采用面向接口的编程技术将逻辑层和数据库访问层进行隔离。任务 8-4 实现订单处理和订单查询流程。

2. 数据库表设计分析

订单由客户的配送地址信息及商品信息组成。配送地址信息比较单一，而商品信息由多条记录组成。因此，设计表时可以使用两张表来描述。一张表描述客户配送地址等基本信息，定义为订单基本表，命名为 t_order。另一张表描述订单商品信息，定义为订单商品项详细表，命名 t_orderdetail。订单基本表和订单详细表是主辅表的关系，通过订单号进行关联。两张表的结构如表 8-1 和表 8-2 所示。

表 8-1 订单基本表 t_order

表名：t_order(订单基本表)					
列 名	描 述	数据类型(精度范围)	空/非空	唯一	约 束 条 件
OrderID	订单编号	int	否	是	主键自增，种子值 100001
UserName	用户登录名	Varchar(20)	否	否	为用户注册账户表中的主键
OrderDate	订购日期	datetime	是	否	
RealName	用户姓名	Varchar(50)	是	否	
Address	配送地址	Varchar(50)	是	否	
ZipCode	邮编	Char(6)	是	否	6 为数字
Phone	电话	Varchar(50)	是	否	
E-mail	电子邮件	Varchar(100)	是	否	
CompleteDate	处理完毕时间	datetime	是	否	
OrderState	订单状态	Char(1)	是	否	“1”：未处理，“2”：处理完毕
其他说明	Primary Key ：OrderID；Foreign Key：UserName				

表 8-2 订单商品项详细表 t_orderdetail

表名：t_orderdetail(订单详细表)					
列 名	描 述	数据类型(精度范围)	空/非空	唯一	约 束 条 件
OrderID	订单编号	int	否	否	与 BookID 构成双主键约束，为 t_order 表中的主键
BookID	图书编号	int	否	否	与 OrderID 构成双主键约束
BookName	书名	Nvarchar(50)	否	否	
UnitPrice	单价	money	否	否	
Quantity	数量	int	否	否	
其他说明	Primary Key ：OrderID、BookID；Foreign Key：OrderID、BookID				

8.3 相关知识

项目 8 中涉及下面几个主要的知识点：三层应用程序架构、Wizard 控件、Panel 控件、用户自定义控件和 ICollection 接口。前面已经对三层应用程序架构技术做了分析，下面对 Wizard 控件、Panel 控件、用户自定义控件和 ICollection 接口的使用进行介绍。

1. Wizard 控件的使用

通过使用窗体收集用户输入是 Web 开发中一个要反复涉及的任务。用来完成某个任务的一组窗体通常称为“向导”。生成一系列相互连接的窗体来分解数据的收集工作是一种普遍的做法。可以通过在每个步骤中管理各窗体之间的导航、数据持久性和状态管理来做到这一点。这样的做法一般比较复杂。ASP.NET 提供了 Wizard 控件,可以简化许多与生成一系列窗体以收集用户输入的操作关联的任务。Wizard 控件提供了一种简单的机制,允许轻松地生成步骤、添加新步骤或重新安排步骤。无须编写代码即可生成线性和非线性的导航,并自定义控件的用户导航。

利用 Wizard 控件,可以使用分离的步骤来收集数据,这样就允许用户在各步骤之间自主导航,从而获得更简单的用户体验。作为一名开发人员,不必担心如何跨页保存数据的问题,Wizard 控件会在用户完成各个步骤时维护好状态。

Wizard 控件的向导使用多个来描绘用户数据输入的不同部分。向导可以根据需要带有任意数量的中间步骤。可以添加不同的控件(如 TextBox 或 ListBox 控件)来收集用户输入。下面的代码示例演示了带有两个步骤的 Wizard 控件。

```
<asp:Wizard ID = "Wizard1" Runat = "server">
    <WizardSteps>
        <asp:WizardStep Runat = "server" Title = "Step 1">
        </asp:WizardStep>
        <asp:WizardStep Runat = "server" Title = "Step 2">
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

两个步骤都包含在 WizardSteps 中,每个步骤中都定义了 Title 属性,该属性用于设置每个步骤功能的文字标题。在每个步骤中都可以添加控件和标签,并可接受用户数据。Wizard 控件可帮助管理要显示哪个步骤以及维护所收集的数据。

Wizard 控件的向导导航功能可以实现线性导航和非线性导航的功能。该控件的状态管理功能允许用户在各个步骤之间前后移动,向导导航功能的各个步骤可以显示在侧栏,允许用户选择导航步骤。属性 StepNextButtonText、StepPreviousButtonText 和 FinishCompleteButtonText,可以自定义用于导航的文本。

```
<asp:Wizard  ID = "Wizard1" Runat = "server"
             StepNextButtonText = " 下一步"
             StepPreviousButtonText = " 上一步"
             FinishCompleteButtonText = " 完成">
```

Wizard 控件内的每个步骤均会给定一个 StepType,用以指示这一步骤是开始步骤、中间步骤还是完成步骤。其属性列表见表 8-3。

Wizard 控件可自动显示标题和控件的当前步骤。标题是用 HeaderText 属性自定义的。可以通过使用 HeaderTemplate 属性来调整标题的模板。属性 ActiveStepIndex 显示当前是向导中的第几个步骤,在页面刚开始加载时,默认是 0。属性 DisplaySideBar 控制是否显示侧边栏,当该属性设置为 True 时,则将整个流程的步骤全部显示在页面中。属性

表 8-3　Wizard 控件的 StepType 属性列表

属性值	说　明
Auto	根据步骤在集合中的顺序，自动显示相关的导航按钮。该项为控件默认值
Start	该步骤是要显示的第一个步骤。不呈现“上一步”按钮
Step	该步骤是介于第一个步骤和最后一个步骤之间的任意步骤。呈现用于导航的“上一步”和“下一步”按钮
Finish	该步骤是收集用户数据的最后一个步骤。呈现用于导航的“完成”按钮
Complete	该步骤是要显示的最后一个步骤。不呈现任何导航按钮

DisplayCancelButton 可以为每个步骤的页面增加一个取消按钮，当该属性设置为 True 时，在每个页面中，都将显示一个 Cancel 按钮，要处理取消的事件，可以在 CancelButtonClick() 中编写代码。

Wizard 控件的 NavigationButtonStyle 属性提供了一种将所有按钮设置为通用样式的简单方法，同时又提供了单独自定义各个按钮的灵活性。NavigationButtonStyle 属性可应用于呈现的所有按钮。但是，可以通过设置个别按钮的样式属性来重写此样式。

Wizard 控件的向导模板可以通过 StartNavigationTemplate、FinishNavigationTemplate、StepNavigationTemplate 和 SideBarTemplate 属性来进一步自定义该控件的界面。

Wizard 控件具有一些事件，可以通过使用自定义代码和事件来自定义 Wizard 控件的行为。例如，可以截获 NextButtonClick 事件，此事件在用户单击“下一步”按钮时引发并捕获当前步骤的数据。传递此事件的 WizardNavigationEventArgs 参数包括 CurrentStepIndex 和 NextStepIndex 属性，能够基于当前步骤和后续步骤来自定义控件的行为，或在用户单击“下一步”按钮时取消导航。主要事件如表 8-4 所示。

表 8-4　Wizard 控件事件列表

名　称	说　明
ActiveStepChanged	当从一个步骤转换到另一个步骤时触发的事件
PreviousButtonClick	当单击“上一步”按钮时触发的事件
NextButtonClick	当单击“下一步”按钮时触发的事件
FinishButtonClick	当单击“完成”按钮时触发的事件
CancelButtonClick	当单击“取消”按钮时触发的事件
SideBarButtonClick	当单击侧栏按钮时触发的事件

2. Panel 控件的使用

Panel Web 服务器控件可以作为静态文本和其他控件的父级控件使用。对于一组控件和相关的标记，可以通过把其放置在 Panel 控件中，然后操作此 Panel 控件的方式将它们作为一个单元进行管理。例如，可以通过设置面板的 Visible 属性来隐藏或显示该面板中的一组控件。

Panel 控件与默认按钮关联，具有单击默认按钮的效果。可将 TextBox 控件和 Button 控件放置在 Panel 控件中，然后通过将 Panel 控件的 DefaultButton 属性设置为面板中某个按钮的 ID 来定义一个默认的按钮。如果用户在面板内的文本框中进行输入时按 Enter 键，这与用户单击特定的默认按钮具有相同的效果。这有助于用户更有效地使用项目窗体。

Panel 控件可以为其他控件添加滚动条。有些控件(如 TreeView 控件)没有内置的滚动条。通过在 Panel 控件中放置滚动条控件,可以添加滚动行为。若要向 Panel 控件添加滚动条,要设置 Height 和 Width 属性,将 Panel 控件限制为特定的大小,然后再设置 ScrollBars 属性。

可使用 Panel 控件在网页上创建具有自定义外观和行为的区域,如创建一个带标题的分组框时,可设置 GroupingText 属性来显示标题。呈现网页时,Panel 控件的周围将显示一个包含标题的框,其标题是用户指定的文本。Panel 控件支持外观属性(如 BackColor 和 BorderWidth),如要在网页上创建具有自定义颜色或其他外观的区域,可以设置外观属性为网页上的某个区域,创建独特的外观。

3. 用户控件的使用

除了使用 HTML 和 Web 服务器控件以外,有时可能需要创建自定义的可重用控件,或者使自己的控件中具有内置 Web 服务器控件未提供的功能。在这种情况下,可以创建自己的控件。这些控件称为用户控件。用户控件是能够在其中放置标记和 Web 服务器控件的容器。然后,可以将用户控件作为一个单元对待,为其定义属性和方法。

(1) 用户控件结构

ASP.NET Web 用户控件与完整的 ASP.NET 网页(.aspx 文件)相似,同时具有用户界面页和代码。可以采取与创建 ASP.NET 网页相似的方式创建用户控件,然后向其中添加所需的标记和子控件。用户控件可以像页面一样包含对其内容进行操作(包括执行数据绑定等任务)的代码。

用户控件与 ASP.NET 网页有以下区别:用户控件的文件扩展名为 .ascx;用户控件中没有 @ Page 指令,而是包含 @ Control 指令,该指令对配置及其他属性进行定义;用户控件不能作为独立文件运行,而必须像处理任何控件一样,将它们添加到 ASP.NET 网页中;用户控件中没有 html、body 或 form 元素,这些元素必须位于宿主页中。

可以在用户控件上使用与在 ASP.NET 网页上所用相同的 HTML 元素(html、body 或 form 元素除外)和 Web 控件。例如,如果要创建一个将用作工具栏的用户控件,则可以将一系列 Button Web 服务器控件放在该控件上,并创建这些按钮的事件处理程序。

(2) 注册用户控件

通过在宿主页上进行注册,可以将用户控件添加到页面中。注册用户控件时,要指定包含用户控件的 .ascx 文件、标记前缀以及将用于在页面上声明用户控件的标记名称。

(3) 定义用户控件的属性和方法

可以采用定义页面的属性和方法时所用的方式定义用户控件的属性和方法。通过定义用户控件的属性,就能以声明方式及利用代码设置其属性。

(4) 用户控件中事件

用户控件包含 Web 服务器控件时,可以在用户控件中编写代码来处理其子控件引发的事件。例如,如果用户控件包含一个 Button 控件,则可以在用户控件中为该按钮的 Click 事件创建处理程序。默认情况下,用户控件中的子控件引发的事件对于宿主页不可用。但是,可以为用户控件定义事件并引发这些事件,以便将子控件引发的事件通知宿主页。进行此操作的方式与定义任何类的事件一样。

(5) 引用外部资源

用户控件运行时,会将该用户控件的 URL 作为基 URL,以解析对外部资源(如图像或其他页面的定位点)的引用。例如,如果用户控件包含一个 Image 控件,而此控件的 ImageUrl 属性设置为 Images/Button1. gif,则会将图像的 URL 添加到用户控件的 URL 以解析该图像的完整路径。如果用户控件引用的资源不在用户控件本身的子文件夹中,则必须提供在运行时解析为正确文件夹的路径。

4. ICollection 接口的使用

ICollection 接口定义所有非泛型集合的大小、枚举数和同步方法。ICollection 接口是 System. Collections 命名空间中类的基接口。ICollection 接口扩展 IEnumerable; IDictionary 和 IList 则是扩展 ICollection 的更为专用的接口。IDictionary 实现是键/值对的集合。IList 实现是值的集合,其成员可通过索引来访问。如果 IDictionary 接口和 IList 接口都不能满足所需集合的要求,则从 ICollection 接口派生新集合类以提高灵活性。

ICollection 接口的方法和属性如表 8-5 和表 8-6 所示。

表 8-5 ICollection 接口方法

名　称	说　明
CopyTo	从特定的 Array 索引处开始,将 ICollection 的元素复制到一个 Array 中
GetEnumerator	返回一个循环访问集合的枚举数(继承自 IEnumerable)

表 8-6 ICollection 接口属性

名　称	说　明
Count	获取 ICollection 中包含的元素数
IsSynchronized	获取一个值,该值指示是否同步对 ICollection 的访问(线程安全)
SyncRoot	获取可用于同步 ICollection 访问的对象

8.4 项目实施

下面将介绍项目的实施过程。实施过程共有 4 个任务组成。图书订购流程在整个项目实施过程中是比较复杂的流程,分解为 3 个任务完成,任务 8-1～任务 8-3 完成图书订购流程的设计。订单处理和订单查询流程相对简单,由任务 8-4 完成。

8.4.1 任务 8-1 使用用户控件设计图书订购流程页面

1. 任务介绍

网上书店的图书购买流程至少包含 4 个以上的步骤。第一步产生购物清单,该环节的购物清单可以修改,一般可以通过购物车实现。产生购物清单后,进入第二步,填写配送地址。地址填写无误,进入第三步,对用户填写的地址信息和购买的商品信息进行确认。到目前为止用户填写的配送地址信息和购买的商品信息都可以修改,当用户确认无误后,进入最

后一步产生订单,用户订单信息写入到数据库,并返回一个订单号,不可以再更改订单信息。复杂一点的购物流程在第二步填写配送地址之后会增加一步,要求客户填写付款方式,以便在线付款。本任务仅讨论图书购买流程的 4 个主要环节,不考虑在线付款。下面介绍 4 个主要流程的页面设计。

2. 任务分析

页面设计主要是完成应用程序架构设计中的表示层的设计。表示层的主要任务是与用户交互。用户通过表示层将数据库递交给逻辑层处理,或从逻辑层获取数据通过表示层展示。Web 表示层(Web Component)模块与业务逻辑层(BLL)模块之间的关系如图 8-2 所示。

多个页面之间有些区域是可以共用的,或者同一个页面中各个区域的功能是不一样的,为使部分页面可以重用,或降低页面的复杂度,可以使用用户控件。本任务中针对购物流程的四个阶段分别设计 4 个用户控件:购物清单界面控件、配送地址录入界面控件、订单清单界面控件和购买成功生成订单号的界面控件。在下面的步骤中将对这四个描述界面的用户控件设计过程分别介绍。

图 8-2　表示层的模块结构图

(1) 设计购物清单操作界面

购物清单界面设计主要是购物车的设计,界面如图 8-3 所示。项目 7 中已经对购物车设计的详细过程做了介绍,这里不再重复介绍。本步骤的主要任务是将项目 7 中的购物车界面转换为用户控件。用户控件包括控件的界面和与界面元素相关的事件。

清除	图书编号	图书名称	销售价格	价格小计	更新数量
清除	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
清除	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
清除	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
清除	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
清除	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
					更新

[lblTotalPrice]
[lblMsg]

图 8-3　购物清单界面

用户控件文件的后缀为. asxc,如图 8-3 所示购物清单用户控件文件可以命名为 CartControl. ascx,控件的界面文件所对应的事件处理程序可以存储在. cs 文件中,一般以界面文件名加. cs 作为事件处理程序文件名。如购物清单用户控件的事件处理程序文件名为 CartControl. ascx. cs。使用@ Control 指令对用户控件配置及其属性进行定义。用户控件中没有 html、body 或 form 元素,这些元素在包含用户控件的宿主页面中。购物清单的用户控件定义的页面代码如下:

```
<%@ Control Language = "C#" AutoEventWireup = "true"
            CodeFile = "CartControl.ascx.cs" Inherits = "Controls_CartControl" %>
<asp:GridView ID = "gvCart"   runat = "server"
```

```
            DataKeyNames = "BookID"
            ShowFooter = "True"
            OnSelectedIndexChanged = "gvCart_SelectedIndexChanged" >
        <Columns>
            <asp:CommandField HeaderText = "清除" SelectText = "清除"
                              ShowSelectButton = "True" >
                <ItemStyle ForeColor = "Blue" />
            </asp:CommandField>
            <asp:BoundField DataField = "BookID" HeaderText = "图书编号" />
            <asp:BoundField DataField = "BookName" HeaderText = "图书名称" />
            <asp:BoundField DataField = "BookPrice" DataFormatString = "{0:c}"
                              HeaderText = "销售价格" />
            <asp:BoundField DataField = "SubTotalPrice" DataFormatString = "{0:c}"
                              HeaderText = "价格小计" />
            <asp:TemplateField HeaderText = "更新数量">
                <ItemTemplate>
                    <asp:TextBox ID = "txbBookQuantity"
                                 runat = "server"
    OnTextChanged = "txbBookQuantity_TextChanged"
                                 Text = '<%# Bind("Quantity") %>'
                                 Width = "50px">
                    </asp:TextBox>
                </ItemTemplate>
                <FooterTemplate>
                    <asp:LinkButton ID = "lbtUpdate" runat = "server"
                                    ForeColor = "Blue">更新</asp:LinkButton>
                </FooterTemplate>
            </asp:TemplateField>
        </Columns>
        <EmptyDataTemplate>购物车没有图书信息!</EmptyDataTemplate>
        </asp:GridView>
    <asp:Label ID = "lblTotalPrice" runat = "server" Width = "200px"/>
    <br />
    <asp:Label ID = "lblMsg" runat = "server" ForeColor = "Red" Width = "200px"/>
```

将项目 7 中的购物车处理事件代码移到用户控件事件处理程序类文件 CartControl.ascx.cs 中即可。

(2) 设计配送地址操作界面

用户选购商品后，要填写配送地址以便网站的安排图书配送。配送地址用户控件设计界面文件名为 AddressControl.ascx。配送地址一般包含用户姓名、配送的详细地址、邮编、联系电话和 E-mail 等基本信息。配送地址界面布局如图 8-4 所示。

配送地址信息需要填写完整，否则无法递送商品，因此，需对用户填写的配送地址信息做必要的验证。从三个方面来进行验证，一是使用验证控件进行验证，如所有的信息不能为空，可以使用必填验证控件进行验证；如果要求输入的数据符合某个规则，可以使用正则验证控件，如邮编和 E-mail。二是可以配置录入控件的属性对输入数据文本的长度进行验证，如限制输入框的文本长度不大于某个数值。三是可以编写事件处理程序进行验证。

用户姓名
请输入用户名!
配送地址
请输入配送地址!
邮编
请输入邮编! 邮编为六位数字字符!
联系电话
请输入联系电话!
E-mail
请输入E-mail! Email格式无效!

图 8-4　配送地址界面

比如输入邮编信息是可以使用 TextBox 控件：

```
<asp:TextBox ID = "txtZip"
             runat = "server"
             MaxLength = "6"
             Width = "150px"/>
```

控件的 MaxLength 属性限定了邮编的长度最多不超过 6 位字符。使用必填验证控件 valZip 进行必填验证，必填验证控件的 ControlToValidate 属性设为输入控件 TextBox 的 ID 值“txtZip”。ErrorMessage 属性值设置的信息“请输入邮编!”将会在用户没有输入邮编时显示。

```
<asp:RequiredFieldValidator ID = "valZip"
                            runat = "server"
                            ControlToValidate = "txtZip"
                            ErrorMessage = "请输入邮编!"/>
```

使用正则验证控件对用户输入的数据进行规则校验。正则验证控件的 ControlToValidate 属性设为输入控件 TextBox 的 ID 值“txtZip”。Display 属性值为“Dynamic”说明验证控件仅在给出出错验证消息时才占位，否则位置将被其他控件填充，如果属性值为“Static”，则页面上验证控件所占的位置将空出来，不被其他控件填充。其他验证的控件的 Display 功能与此类似。ValidationExpression 属性可以设置正则表达式，限制字符输入的格式。限制邮编输入的正则表达式为“\d{6}”，表示只能输入 6 位数字字符。正则表达式的内容这里不再详细讲述，请参考其他教材的内容。

```
<asp:RegularExpressionValidator ID = "valZip1"
                                runat = "server"
                                ControlToValidate = "txtZip"
                                Display = "Dynamic"
                                ErrorMessage = "邮编为六位数字字符!"
                                ValidationExpression = "\d{6}"/>
```

配送地址用户控件其他界面元素不再做详细分析，下面给出界面设计视图控件元素详细代码：

```
<%@ Control Language = "C#" AutoEventWireup = "true"
```

```
            CodeFile = "AddressControl.ascx.cs"
            Inherits = "Controls_AddressControl" %>
<table border = "0" cellpadding = "0" cellspacing = "0" width = "60 %">
    <tr>
        <td valign = "top">
            用户姓名<br />
            <asp:TextBox ID = "txtRealName"
                         runat = "server"
                         MaxLength = "80"
                         Width = "150px"/><br />
            <asp:RequiredFieldValidator ID = "valFirstName"
                                        runat = "server"
                                        ControlToValidate = "txtRealName"
                                        ErrorMessage = "请输入用户名!"/>
        </td>
    </tr>
    <tr>
        <td valign = "top">
            配送地址<br />
            <asp:TextBox ID = "txtAddress"
                         runat = "server"
                         MaxLength = "80"
                         Width = "500px"/><br />
            <asp:RequiredFieldValidator ID = "valAddress"
                                        runat = "server"
                                        ControlToValidate = "txtAddress"
                                        ErrorMessage = "请输入配送地址!"/>
        </td>
    </tr>
    <tr>
        <td valign = "top">
            邮编<br />
            <asp:TextBox ID = "txtZip"
                         runat = "server"
                         MaxLength = "6"
                         Width = "150px"/><br />
            <asp:RequiredFieldValidator ID = "valZip"
                                        runat = "server"
                                        ControlToValidate = "txtZip"
                                        ErrorMessage = "请输入邮编!"/>
            <asp:RegularExpressionValidator ID = "valZip1"
                                    runat = "server"
                                    ControlToValidate = "txtZip"
                                    Display = "Dynamic"
                                    ErrorMessage = "邮编为六位数字字符!"
                                    ValidationExpression = "\d{6}"/>
        </td>
    </tr>
    <tr>
        <td valign = "top">
            联系电话<br />
            <asp:TextBox ID = "txtPhone"
```

```
                        runat = "server"
                        MaxLength = "50"
                        Width = "150px"></asp:TextBox><br />
                <asp:RequiredFieldValidator ID = "valPhone"
                                            runat = "server"
                                            ControlToValidate = "txtPhone"
                                            ErrorMessage = "请输入联系电话!"/>
            </td>
        </tr>
        <tr>
            <td valign = "top">
                Email<br />
                <asp:TextBox ID = "txtEmail"
                        runat = "server"
                        MaxLength = "80"
                        Width = "500px"></asp:TextBox><br />
                <asp:RequiredFieldValidator ID = "valEmail"
                                            runat = "server"
                                            ControlToValidate = "txtEmail"
                                            ErrorMessage = "请输入 Email!"/>
                <asp:RegularExpressionValidator ID = "valEmail1"
                                            runat = "server"
                                            ControlToValidate = "txtEmail"
                                            Display = "Dynamic"
                                            ErrorMessage = "Email 格式无效!"
              ValidationExpression = "\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"/>
            </td>
        </tr>
    </table>
```

配送地址用户控件事件处理程序包含在文件 AddressControl.ascx.cs 中。文件中定义部分类 Controls_AddressControl，继承 System.Web.UI.UserControl 类，类中设置一个属性 Address，数据类型为 AddressInfo，为地址实体类，该类将在本项目的任务 8-3 中进行描述。存储器的 get 访问器中首先对输入文本进行非空验证，只要有一个文本输入为空，则 get 访问器返回空值，然后调用可重用的静态类 WebUtility 的 InputText 方法对输入的文本进行处理。最后使用输入的有效文本作为参数实例化地址对象 AddressInfo，并通过 get 访问器返回。

set 访问器中要求输入的数据为 AddressInfo 类型的对象，首先输入的对象不能为空，然后将对象中的不为空的属性值绑定到地址用户控件的相应 TextBox 控件中。

配送地址用户控件事件处理过程代码如下：

```
public partial class Controls_AddressControl : System.Web.UI.UserControl{
        /// <summary>
        /// 定义获取用户地址的属性
        /// </summary>
        public AddressInfo Address {
          get {
              if (string.IsNullOrEmpty(txtRealName.Text) &&
                 string.IsNullOrEmpty(txtAddress.Text) &&
```

```
                string.IsNullOrEmpty(txtZip.Text) &&
                string.IsNullOrEmpty(txtPhone.Text) &&
                string.IsNullOrEmpty(txtEmail.Text))
                return null;

            string realName = WebUtility.InputText(txtRealName.Text, 80);
            string address = WebUtility.InputText(txtAddress.Text, 80);
            string zipCode = WebUtility.InputText(txtZip.Text, 6);
            string tel = WebUtility.InputText(txtPhone.Text, 50);
            string email = WebUtility.InputText(txtEmail.Text, 80);

            return new AddressInfo(realName, address, zipCode, tel, email);
        }
        set {
            if(value != null) {
                if (!string.IsNullOrEmpty(value.RealName))
                    txtRealName.Text = value.RealName;
                if (!string.IsNullOrEmpty(value.Address))
                    txtAddress.Text = value.Address;
                if (!string.IsNullOrEmpty(value.ZipCode))
                    txtZip.Text = value.ZipCode;
                if (!string.IsNullOrEmpty(value.Phone))
                    txtPhone.Text = value.Phone;
                if (!string.IsNullOrEmpty(value.Email))
                    txtEmail.Text = value.Email;
            }
        }
    }
}
```

WebUtility 类是一个可重用的类，用于对界面输入的文本进行处理，将界面输入中影响数据库安全的因素过滤掉，保证进入数据库的数据是有效的数据。WebUtility 类包含在独立文件 WebUtility.cs 中。对文本进行处理时使用了正则表达式类，需要引入命名空间 System.Text.RegularExpressions。

引入命名空间的代码如下：

```
public static class WebUtility{
    public static string InputText(string text, int maxLength){
        text = text.Trim();
        //如果用户输入的文本为 null 或空，则返回空字符串
        if (string.IsNullOrEmpty(text))
            return string.Empty;
        //如果输入的文本大于给定的最大长度，则按最大长度截取文本
        if (text.Length > maxLength)
            text = text.Substring(0, maxLength);
        //当文本中连续出现多个空字符时，则用 1 个空字符替换
        text = Regex.Replace(text, "[\\s]{2,}", " ");
        //将文本中的"<br/>" 换行标识或者"<p>"分段替换为"\n"
        text = Regex.Replace(text, "(<[b|B][r|R]/*>)+|(<[p|P](.|\\n)*?>)", "\n");
        //将文本中的描述空格的实体定义“ ”替换为空格
        text = Regex.Replace(text, "(\\s*&[n|N][b|B][s|S][p|P];\\s*)+", " ");
```

```
            //将文本中的<(.|\\n)>替换为空字符
            text = Regex.Replace(text, "<(.|\\n) * ?>", string.Empty);
            //将输入文本中的“'”替换为“''”
            text = text.Replace("'", "''");
            return text;
        }

        //将文本中的非单词字符替换为空字符。
        public static string CleanNonWord(string text){
            return Regex.Replace(text, "\\W", "");
        }
    }
```

（3）设计订单信息显示界面

配送地址填写好后，要显示完整的订单信息，订单信息的用户控件如图 8-5 所示。界面的上半部分为送货地址，中间部分为购物清单，可以用 GridView 控件呈现购物清单，下半部分为订单的总价。

送货地址：

用户姓名：[Literal "ltlRealName"]
配送地址：[Literal "ltlAddress"]
邮编：[Literal "ltlZip"]
联系电话：[Literal "ltlPhone"]
E-mail：[Literal "ltlEmail"]

购物清单：

图书名称	销售价格	订购数量	价格小计
数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定

总价：[lblTotalPrice]

图 8-5 订单详细信息

在呈现用户地址信息时，使用 Literal 控件，该控件可在无须添加任何 HTML 元素的情况下将静态文本呈现在网页上，并且可以通过服务器代码以编程方式控制文本。

控件界面元素的编码如下：

```
<%@ Control Language = "C#" AutoEventWireup = "true"
              CodeFile = "OrderControl.ascx.cs"
              Inherits = "Controls_OrderControl" %>
<table border = "0" cellpadding = "0" cellspacing = "0">
    <tr>
        <td>
            <asp:Label ID = "Label2"  runat = "server"
                    Font - Bold = "True"  Font - Size = "15px"
                    Text = "送货地址: "/></td>
    </tr>
    <tr>
        <td>
            用户姓名:<asp:Literal ID = "ltlRealName" runat = "server"/><br />
            配送地址:<asp:Literal ID = "ltlAddress" runat = "server"/><br/>
```

```
            邮编:<asp:Literal ID="ltlZip" runat="server"/><br/>
            联系电话:<asp:Literal ID="ltlPhone" runat="server"/><br/>
            Email:<asp:Literal ID="ltlEmail" runat="server"/><br />
         </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="Label1" runat="server"
                        Font-Bold="True" Font-Size="15px"
                        Text="购物清单:"/><br />
            <asp:GridView ID="gvCart" runat="server">
                <Columns>
                    <asp:BoundField DataField="BookName"
                                HeaderText="图书名称" />
                    <asp:BoundField DataField="BookPrice"
                                DataFormatString="{0:c}"
                                HeaderText="销售价格" />
                    <asp:BoundField DataField="Quantity"
                                HeaderText="订购数量" />
                    <asp:BoundField DataField="SubTotalPrice"
                                DataFormatString="{0:c}"
                                HeaderText="价格小计" />
                </Columns>
            </asp:GridView>
            <br />总价:
            <asp:Label ID="lblTotalPrice" runat="server" Width="200px"/>
        </td>
    </tr>
</table>
```

控件对应的事件编码中设计 3 个 set 属性访问器为用户界面提供数据。第一个属性访问器为地址类型,将读取到的地址信息绑定到控件的地址部分 Literal 控件呈现。代码如下:

```
public AddressInfo Address{
    set{
        if (value != null){
            if (!string.IsNullOrEmpty(value.RealName))
                ltlRealName.Text = value.RealName;
            if (!string.IsNullOrEmpty(value.Address))
                ltlAddress.Text = value.Address;
            if (!string.IsNullOrEmpty(value.ZipCode))
                ltlZip.Text = value.ZipCode;
            if (!string.IsNullOrEmpty(value.Phone))
                ltlPhone.Text = value.Phone;
            if (!string.IsNullOrEmpty(value.Email))
                ltlEmail.Text = value.Email;
        }
    }
}
```

第二个属性访问器用于呈现购物清单，类型为 ICollection 接口，使用该接口需要命名空间 System.Collections。购物车内购物清单存储在 ICollection 接口类型的对象中，赋值给属性访问器，可以直接绑定界面 GridView 控件上。代码如下：

```
public ICollection OrderDetails{
    set{
        if (value!= null){
            gvCart.DataSource = value;
            gvCart.DataBind();
        }
    }
}
```

第三个属性访问器为字符串类型，用于接收和读取订单总价。该属性的定义比较简单。代码如下：

```
public string TotalPrice{
    get{
        if (string.IsNullOrEmpty(lblTotalPrice.Text)) return null;
        return lblTotalPrice.Text;
    }
    set {
        if (value != null){
            lblTotalPrice.Text = value;
        }
    }
}
```

(4) 设计订单号生成界面

用户确认订单，订单信息将写入数据库，同时产生订单号，订单号和订购总金额显示给客户。显示订单号和订购总金额的界面如图 8-6 所示。

图 8-6　显示客户订单号和订单总金额

界面文件为 OrderIDShowControl.ascx，设计界面元素的代码如下：

```
<%@ Control Language = "C#" AutoEventWireup = "true"
              CodeFile = "OrderIDShowControl.ascx.cs"
              Inherits = "Controls_OrderIDShowControl" %>
<div>
   <p><asp:Literal ID = "ltlUserName" runat = "server" Text = "尊敬的客户: " /></p>
   <p><asp:Literal ID = "ltlOrderID" runat = "server" Text = "您的订单号: "/></p>
   <p><asp:Literal ID = "ltlTotalPrice" runat = "server" Text = "您应付款金额: " /></p>
   <p><asp:Literal ID = "Literal1" runat = "server"
                  Text = "若您对订单有疑问,请与我们的客服联系!"/></p>
   <p><asp:Literal ID = "literal2" runat = "server" Text = "感谢您的订购!" /></p>
   <p><asp:HyperLink ID = "HyperLink2" runat = "server"
                  NavigateUrl = "../Default.aspx">继续购物</asp:HyperLink></p>
</div>
```

界面文件对应的事件代码：

```
public partial class Controls_OrderIDShowControl :
            System.Web.UI.UserControl
{
    /// <summary>
    /// 设置用户名
    /// </summary>
    public string UserName{
        set {
            if (value != null){
                ltlUserName.Text += value;
            }
        }
    }

    /// 设置订单号
    public string OrderID{
        set{
            if(value != null){
                ltlOrderID.Text += value;
            }
        }
    }

    /// 设置订单总价
    public string TotalPrice{
        set{
            if(value != null){
                ltlTotalPrice.Text += value;
            }
        }
    }
}
```

(5) 使用 Wizard 控件构建具有导航功能的购物流程

本步骤中使用 Wizard 控件将 4 个自定义用户控件组合在一起。使用用户控件，先注册用户控件，在包含 Wizard 控件的页面中引入下面的代码，分别注册购物清单用户控件、配送地址用户控件、订单详细信息用户控件和生成订单号的用户控件。注册用户控件使用@ Register 指令，属性 TagPrefix 定义控件的前缀，属性 TagName 定义控件的标记，属性 Src 指向用户控件的 URL。定义控件的标记形式为“TagPrefix:TagName”，如 NetBookuc:CartControl。

```
<%@ Register TagPrefix="NetBookuc"
             TagName="CartControl"
             Src="~/Controls/CartControl.ascx" %>

<%@ Register TagPrefix="NetBookuc"
             TagName="AddressControl"
             Src="~/Controls/AddressControl.ascx" %>

<%@ Register TagPrefix="NetBookuc"
```

```
              TagName = "OrderControl"
              Src = "~/Controls/OrderControl.ascx" %>

<%@ Register TagPrefix = "NetBookuc"
              TagName = "OrderIDControl"
              Src = "~/Controls/OrderIDShowControl.ascx" %>
```

Wizard 控件的 ID 为“AcctWizard”，属性 ActiveStepIndex 默认值设为 0，则程序运行时默认从第一个界面开始执行。Wizard 控件向导步骤的第一步完成购物清单的设置。首先在 WizardStep1 中放置一个 Panel 控件，Panel 控件的 DefaultButton 属性设为 LinkButton 按钮的 ID 的值 StartNextButton，单击该按钮可以进入第二步的界面，做了这一设置后，当用户完成第一步操作直接单击 Enter 键，相当于单击了“结算本单商品”按钮，可以导航到第二步。需要注意的是 StartNextButton 按钮的 CommandName 属性值要设为 MoveNext。在 Panel 中还需要放置一个 HyperLink 控件，可以导航到商品选购界面，方便用户继续选购商品。使用 NetBookuc:CartControl 定义购物清单用户控件，ID 设为“CartControl1”。设置 WizardStep1 的代码如下：

```
<asp:WizardStep ID = "WizardStep1" runat = "server" Title = "购物清单" >
    <asp:Panel ID = "panStep1" runat = "server" DefaultButton = "StartNextButton" >
        <!-- 使用用户自定义控件 - 购物清单 -->
        <NetBookuc:CartControl id = "CartControl1" runat = "server"/>
        <asp:HyperLink ID = "HyperLink1" runat = "server"
                NavigateUrl = "../Default.aspx">继续购物</asp:HyperLink>
        <asp:LinkButton ID = "StartNextButton" runat = "server"
                CommandName = "MoveNext" Text = "结算本单商品" />
    </asp:Panel>
</asp:WizardStep>
```

Wizard 控件向导步骤的第二步完成配送地址的设置。WizardStep2 中放置一个 Panel 控件，Panel 中定义一个配送地址用户控件 AddressControl1，放置两个 LinkButton 按钮，第一个按钮的 ID 设为“StepPreviousButton”，属性 CommandName 设为“MovePrevious”，文本属性 Text 设为“填写购物清单”，则单击该按钮可以返回到上一步“购物清单界面”。第二个 LinkButton 按钮 ID 属性值设为“StepNextButton”，属性 CommandName 的值为“MoveNext”，属性 Text 的值为“提交订单”，单击该按钮可以导航到第三步。Panel 控件的 DefaultButton 属性与“StepNextButton”关联。设置 WizardStep2 的代码如下：

```
<asp:WizardStep ID = "WizardStep2" runat = "server" Title = "收货人地址">
    <asp:Panel ID = "panStep2" runat = "server" DefaultButton = "StepNextButton">
        <!-- 使用用户自定义控件 - 送货地址 -->
        <NetBookuc:AddressControl ID = "AddressControl1" runat = "server" />
          <asp:LinkButton ID = "StepPreviousButton" runat = "server"
                  CausesValidation = "False"
                  CommandName = "MovePrevious"
                  Text = "填写购物清单" />
          <asp:LinkButton ID = "StepNextButton" runat = "server"
                  CommandName = "MoveNext"
                  Text = "提交订单" />
```

```
        </asp:Panel>
    </asp:WizardStep>
```

Wizard 控件向导步骤的第三步完成订单详细信息的显示设计。第三步中设置 WizardStep 的属性 StepType="Finish",表示该步骤确认后,数据将写入数据库,不可更改。与前两个步骤设置方法一样,在 WizardStep3 中先放置一个 Panel 控件。Panel 中定义用户控件 OrderControl1,显示地址的详细信息。此外,Panel 中还需放置两个 LinkButton 控件,第一个按钮可以导航到前一步,第二个按钮完成购物,产生订单,并导航到第四步。代码设置如下:

```
<asp:WizardStep ID="WizardStep3" runat="server"
                Title="提交订单" StepType="Finish">
    <asp:Panel ID="panStep3" runat="server" DefaultButton="FinishButton" >
      <!-- 使用用户自定义控件 - 订单确认 -->
      <NetBookuc:OrderControl ID="OrderControl1"  runat="server" />
      <asp:LinkButton ID="FinishPreviousButton" runat="server"
                   CausesValidation="False"
                   CommandName="MovePrevious"
                   Text="填写收货人地址" />
      <asp:LinkButton ID="FinishButton" runat="server"
                   CommandName="MoveComplete"
                   Text="确认提交订单" />
      </asp:Panel>
</asp:WizardStep>
```

导航步骤的最后一步中,WizardStep 的属性 StepType="Complete",AllowReturn="False",完成整个导航步骤的设置,并且不可返回。使用标记"NetBookuc:OrderIDControl"定义用户控件"OrderIDControl1",呈现用户订单号和购物金额。代码如下:

```
<asp:WizardStep ID="WizardStep4" runat="server"
               AllowReturn="False"  Title="订购完成"
               StepType="Complete">
     <!-- 使用用户自定义控件 - 订单确认 -->
     <NetBookuc:OrderIDControl ID="OrderIDControl1"  runat="server" />
 </asp:WizardStep>
```

可以在 Wizard 控件的 HeaderTemplate 模板中显示导航步骤的标题,使用如下的代码:

```
<HeaderTemplate>
      <% = AcctWizard.ActiveStep.Title %>
</HeaderTemplate>
```

本任务中 Wizard 可以定义两个事件分别响应控件界面上的 3 个按钮动作。步骤每改变一次触发 ActiveStepChanged 事件,单击最后一个步中的"完成"按钮触发 FinishButtonClick 事件。下面分别介绍这两个事件的处理过程。

事件中要对操作用户进行登录验证,登录验证过程可以写成一个函数,验证函数代码如下:

```
private void IsAuthenticated(){
        //如果用户没有登录则导向登录页面,要求用户登录
        if (!User.Identity.IsAuthenticated){
            Response.Redirect("~/Web/Logon.aspx");
        }
    }
```

当 Wizard 控件的向导步骤每改变一次,便执行一次 ActiveStepChanged 事件。使用控件的属性 ActiveStepIndex 可以获取当前步骤的索引。当索引为 0 时,处于订购流程的第一步“操作购物清单界面”,允许匿名用户操作购物车,不需要进行登录验证,其他步均需要登录验证。在订购操作的过程中,如果用户的购物车为空,则用户不能完成订购任务。当索引值为 1 时,处于购物流程第二步“填写配送地址界面”,如果用户的配送地址为空,则获取用户的注册地址作为配送地址。当索引值为 2 时,将配送地址信息和购物车商品详细信息绑定到订单用户控件显示。当索引值为 3 时,完成订购任务,产生订单,显示用户名、订单号和订单总价格。ActiveStepChanged 事件程序代码如下:

```
protected void AcctWizard_ActiveStepChanged(object sender, EventArgs e){
        if (AcctWizard.ActiveStepIndex != 0){
            //登录验证
            IsAuthenticated();

            //填写配送地址和确认订单阶段需要判断购物车是否为空
            if (Profile.ShoppingCart.CartItems.Count <= 0 &&
            AcctWizard.ActiveStepIndex < 3){
                Response.Redirect("~/Web/FinishedOrder.aspx?lblMsg=
            不能订购,您购物车为空!");
            }
        }

        if (AcctWizard.ActiveStepIndex == 1){
            //若用户配送地址信息为空,则读取用户注册地址信息作为配送地址
            if(AddressControl1.Address == null){
                AddressControl1.Address = AccountInfo.GetAccountInfo(Profile.UserName);
            }
        }
        else if (AcctWizard.ActiveStepIndex == 2){
            //从 proflie 中获取购物清单信息绑定到用户自定义订单显示界面
            //地址信息从配送地址用户控件中获取
            OrderControl1.Address = AddressControl1.Address;
            OrderControl1.OrderDetails = Profile.ShoppingCart.CartItems;
            OrderControl1.TotalPrice = Profile.ShoppingCart.TotalPrice.ToString("c");
        }
        else if (AcctWizard.ActiveStepIndex == 3){
            //显示用户名
            OrderIDControl1.UserName = Profile.UserName;
            //显示订单号
            ltlOrderID.Text = ltlOrderID.Text + "<font color='red'
            size='5'>" + Profile.OrderID + "</font>";
            //显示订单总价格
```

```
            OrderIDControl1.TotalPrice = "<font color='red' size='5'>" +
            OrderControl1.TotalPrice + "</font>";
        }
    }
```

在 FinishButtonClick 事件中完成订单提交事务，当订单提交成功返回订单号，并清空购物车。在产生订单号时调用 Order 类的 InsertOrder 方法，完成订单添加到数据库并获取订单号的方法，Order 类将在下面的步骤中分析。InsertOrder 方法需要一个类型为 OrderInfo 的参数，事件代码中通过一个 SetOrder() 函数，创建订单对象并赋值，作为 InsertOrder 方法的参数，完成产生订单的任务。FinishButtonClick 事件程序代码和 SetOrder 函数的代码如下：

```
protected void AcctWizard_FinishButtonClick(object sender,
            WizardNavigationEventArgs e){
        //登录验证
        IsAuthenticated();

        //购物车不为空，确认提交订单
        if (Profile.ShoppingCart.CartItems.Count>0){
            //订单添加到数据库，并返回订单号
            orderID= Order.InsertOrder(SetOrder());

            //若订购成功，则清空购物车
            if (orderID>0){
               Profile.ShoppingCart.Clear();
            }
        }
}


private OrderInfo SetOrder() {
        OrderInfo userOrder = new OrderInfo();
        if (AddressControl1.Address!= null && Profile.ShoppingCart != null) {
            //填写订单客户信息
            userOrder.UserName = Profile.UserName;
            userOrder.OrderDate = DateTime.Now;
            userOrder.ShippingAddress = AddressControl1.Address;
            userOrder.OrderItems = new
            OrderItemInfo[Profile.ShoppingCart.CartItems.Count];
            int i = 0;

            //将购物车中商品信息迁移到订单对象中
            foreach (CartItem cartItem in Profile.ShoppingCart.CartItems) {
                userOrder.OrderItems[i] = new
            OrderItemInfo(cartItem.BookID, cartItem.BookName,
            cartItem.Quantity, cartItem.BookPrice);
                i++;
            }
        }
        return userOrder;
}
```

购物流程的操作过程如图 8-7～图 8-10 所示。



图 8-7　购物清单



图 8-8　配送地址





图 8-9　订单提交



图 8-10　产生订单

3. 任务完成总结

本任务使用 Wizard 控件完成图书订购流程界面设计，订购流程的每一步骤的界面使用用户控件完成，采用向导控件结合用户控件设计方法，既保证了设计流程的连贯一致，又不至于造成单个设计文件的过长。在流程的最后一步由用户确认，完成订单的生成。订单信息写入数据库，成为有效信息。

4. 课堂训练与知识拓展

用户在网站注册账户时，分几个步骤完成。先填写用户号和密码，再填写基本信息，最后填写详细信息。用户注册过程可以使用向导来完成。使用 Wizard 控件设计用户的注册流程界面。

8.4.2 任务 8-2 设计数据库访问重用类

1. 任务介绍

业务系统实现对数据库的访问可以总结为一个查询、三个操作。查询是指从数据库中读取信息，三个操作是指向数据库中添加数据、更新数据库中的数据、删除数据库中的数据。数据库的这 4 种访问方式可以封装在一个模块中，为所有的业务逻辑访问数据库提供代码复用。本任务将介绍数据库访问重用组件的设计。

2. 任务分析

数据库访问组件在三层软件架构技术中属于数据库访问层，直接构建与数据库的通信。这里主要讨论 SQLServer 数据库的访问。ASP.NET 中实现对数据库的访问需要引入命名空间 System. Data. SqlClient，该命名空间封装了对 SQLServer 数据库的访问类，如 SqlConnection、SqlCommand、SqlDataReader、SqlParameter 等，这些丰富的类满足对数据库的多种访问方式。访问数据库需要提供数据库连接词，连接词在 Web. Config 文件中配置，项目 7 中已经定义了连接词“LocalSqlServer”。读取连接词要用到命名空间 System. Configuration 下的 ConfigurationManager 类，实现代码如下：

```
public static readonly string ConnectionStringLocalTransaction =
ConfigurationManager.ConnectionStrings["LocalSqlServer"].ConnectionString;
```

读取到的连接词存放在一个静态的只读类型的字符串中随时供使用。

本任务中的数据库访问重用类来源于微软所提供的数据库访问助手 SqlHelper，SqlHelper 类封装了多种数据库访问方法，这里将依据项目的需要讨论主要的方法。SqlHelper 类结构如下：

```
public abstract class SqlHelper{
        //从 Web.Config 中获取访问数据库的连接词
        public static readonly string ConnectionStringLocalTransaction =
   ConfigurationManager.ConnectionStrings["LocalSqlServer"].ConnectionString;
```

```
        //定义 HashTable 缓存访问数据库的参数
        private static Hashtable parmCache =
                                  Hashtable.Synchronized(new Hashtable());

        //使用数据连接词连接数据库
        //执行 SQL 语句或存储过程实现对数据库的操作
        public static int ExecuteNonQuery(string connectionString,
                                  CommandType cmdType,
                                  string cmdText,
                                  params SqlParameter[] commandParameters){ }

        //使用连接对象连接数据库
        //执行 SQL 语句或存储过程实现对数据库的操作
        public static int ExecuteNonQuery(SqlConnection connection,
                                  CommandType cmdType,
                                  string cmdText,
                                  params SqlParameter[] commandParameters){ }


        //使用事务处理实现对数据库的操作
        public static int ExecuteNonQuery(SqlTransaction trans,
                                  CommandType cmdType,
                                  string cmdText,
                                  params SqlParameter[] commandParameters){ }

        //查询数据库,返回 SqlDataReader 类型的结果集
                public static SqlDataReader ExecuteReader(string connectionString,
                                  CommandType cmdType,
                                  string cmdText,
                                  params SqlParameter[] commandParameters){ }

        //执行 Sql 语句返回结果集中第一行的第一列
        public static object ExecuteScalar(string connectionString,
                                  CommandType cmdType,
                                  string cmdText,
                                  params SqlParameter[] commandParameters){ }

        //缓存参数列表
        public static void CacheParameters(string cacheKey,
                                  params SqlParameter[] commandParameters){ }

        //读取参数列表
        public static SqlParameter[] GetCachedParameters(string cacheKey){ }

        //数据库访问操作参数预处理
        private static void PrepareCommand(SqlCommand cmd,
                             SqlConnection conn, SqlTransaction trans,
                             CommandType cmdType,
                             string cmdText, SqlParameter[] cmdParms) { }
}
```

SqlHelper 类为抽象类，封装的方法为静态方法，这些方法可供上层处理逻辑直接调用。接下来的步骤中将讨论主要方法的实现。

(1) 实现操作数据库的 ExecuteNonQuery 方法

数据的访问操作需要预先准备参数，并判断是否需要事务处理操作。操作过程通过 SqlHelper 类的私有静态方法 PrepareCommand 完成。代码如下：

```
private static void PrepareCommand(SqlCommand cmd, SqlConnection conn, SqlTransaction trans,
        CommandType cmdType, string cmdText, SqlParameter[] cmdParms) {

        //如果连接对象没有打开,则打开连接对象
        if (conn.State != ConnectionState.Open)
            conn.Open();

        //为 SqlCommand 类型的对象实例 cmd 配置连接对象和 Sql 参数
        //Sql 参数 cmdText 可以为 SQL 语句,也可以是存储过程名称
        cmd.Connection = conn;
        cmd.CommandText = cmdText;

        //如果需要事务处理,则配置事务处理参数
        if (trans != null)
            cmd.Transaction = trans;

        //配置访问数据方式的参数(使用 Sql 访问,还是使用存储过程访问)
        cmd.CommandType = cmdType;

        //配置访问数据库的参数列表
        if (cmdParms != null) {
             foreach (SqlParameter parm in cmdParms)
                 cmd.Parameters.Add(parm);
        }
}
```

使用连接词访问数据库，实现数据库的操作可以使用方法 ExecuteNonQuery 实现。方法 ExecuteNonQuery 有 4 个参数：第一个参数 connectionString 是数据库连接词，可以从 Web.Config 文件中读取；第二个参数 cmdType 定义了访问数据库的方式，如果值是 CommandType.Text，则使用 SQL 语句访问数据库，如果值是 CommandType.StoredProcedure，则使用存储过程访问数据库；第三个参数 cmdText 为 SQL 语句字符串或存储过程名称，具体情况依据参数 cmdType 决定；最后一个参数 commandParameters 为访问数据库的参数数组。代码如下：

```
public static int ExecuteNonQuery(string connectionString,
                        CommandType cmdType,
                        string cmdText,
                        params SqlParameter[] commandParameters){
    SqlCommand cmd = new SqlCommand();
    using (SqlConnection conn = new SqlConnection(connectionString)){
      PrepareCommand(cmd, conn, null, cmdType,
```

```
                                 cmdText,commandParameters);
        int val = cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
        return val;
       }
   }
```

上述代码中使用 using 语句创建 conn 对象，当 using 语句执行完毕后，自动释放连接对象。方法的返回值为影响数据库表的记录行数，即成功操作的记录数。取得操作成功的记录数后，使用 cmd.Parameters.Clear()语句清空参数列表。

（2）实现从数据库中读取数据记录的 ExecuteReader 方法

SqlHelper 类的方法 ExecuteReader 实现从数据库中快速读取数据，方法的返回值为 SqlDataReader 类型。该方法的参数与 SqlHelper 类的 ExecuteNonQuery 方法参数完全一样。方法中使用 cmd.ExecuteReader(CommandBehavior.CloseConnection)实现对数据的读取，并创建 DataReader 对象保存结果。参数 CommandBehavior.CloseConnection 的作用是关闭 DataReader 对象的同时自动关闭数据库连接对象。此外采用 try/ catch 结构，当数据库访问出现异常时，不能通过 CommandBehavior.CloseConnection 关闭连接对象，需要在异常处理中关闭连接对象。代码如下：

```
public static SqlDataReader ExecuteReader(string connectionString,
                                 CommandType cmdType,
                                 string cmdText,
                                 params SqlParameter[] commandParameters)
        {
    SqlCommand cmd = new SqlCommand();
    SqlConnection conn = new SqlConnection(connectionString);

    try {
        PrepareCommand(cmd, conn, null, cmdType,
                    cmdText,commandParameters);
        SqlDataReader rdr =
                cmd.ExecuteReader(CommandBehavior.CloseConnection);
        cmd.Parameters.Clear();
        return rdr;
    }
    catch {
        conn.Close();
        throw;
    }
}
```

（3）执行 SQL 语句返回结果集中第一行的第一列

ExecuteScalar 方法执行一个 SQL 命令返回结果集的第一行的第一列的值。它经常用来执行 SQL 的 COUNT、AVG、MIN、MAX 和 SUM 函数，这些函数都是返回单行单列的结果集。ExecuteScalar 一般返回一个 Object 类型，因此必须把它转换为强类型。如果转换不正确，.NET 框架就会引发 InvalidCastException 异常。代码如下：

```
public static object ExecuteScalar(string connectionString,
                                   CommandType cmdType,
                                   string cmdText,
                                   params SqlParameter[] commandParameters) {
    SqlCommand cmd = new SqlCommand();

    using (SqlConnection connection = new SqlConnection(connectionString)) {
        PrepareCommand(cmd, connection, null, cmdType, cmdText,
                       commandParameters);
        object val = cmd.ExecuteScalar();
        cmd.Parameters.Clear();
        return val;
    }
}
```

(4) 实现参数缓存和参数提取

使用带参数的 SQL 语句或存储过程访问数据库时，可以预先对参数进行缓存，在需要时再从缓存中读取。方法 CacheParameters 将参数数组保存到哈希结构类型的变量中缓存。方法 GetCachedParameters 从哈希变量中读取参数数组。代码如下：

```
public static void CacheParameters(string cacheKey,
                                   params SqlParameter[] commandParameters) {
      parmCache[cacheKey] = commandParameters;
}

public static SqlParameter[] GetCachedParameters(string cacheKey) {
      SqlParameter[] cachedParms = (SqlParameter[])parmCache[cacheKey];

      if (cachedParms == null)  return null;

      SqlParameter[] clonedParms = new SqlParameter[cachedParms.Length];
      //将哈希变量中的参数数组复制到 clonedParms 数组中
      for (int i = 0, j = cachedParms.Length; i < j; i++)
          clonedParms[i] =
            (SqlParameter)((ICloneable)cachedParms[i]).Clone();
      //返回复制的参数数组
      return clonedParms;
}
```

3. 任务完成总结

本任务主要讨论了数据库访问重用组件的设计，数据访问类型主要有两类：一类是数据库查询；另一类是数据库操作。数据库查询可以返回整个结果集，也可以返回结果集第一行的第一列。数据库操作主要是针对数据库表进行添加、删除和更新操作。在进行类设计时，将查询分为两个方法来实现，操作使用一个方法实现。此外在使用上述 3 个方法访问数据库时，可能需要参数列表，在设计类时，还设计了缓存和读取参数列表的两个方法。

4. 课堂训练与知识拓展

在设计数据库查询和操作方法时，除了将数据库连接词作为参数，也可以将连接对象或

事务处理对象作为参数。请重载 ExecuteNonQuery 方法分别实现连接对象和事务处理对象访问数据库的方法。

8.4.3　任务 8-3　订购流程业务处理

1. 任务介绍

订购流程的业务处理主要实现订单业务逻辑层和数据访问层的处理。订单数据由表示层传递到逻辑层后，经过逻辑层的处理，然后调用数据访问层的模块完成与数据库的交互。本任务中主要介绍订购流程业务逻辑层和数据访问层的模块设计，以及模块与模块之间的接口设计。

2. 任务分析

逻辑层和数据访问层模块结构图如图 8-11 所示。业务逻辑层不直接使用数据访问层中的方法，逻辑层与数据访问层之间的通信通过接口来完成，逻辑层 BLL 使用 DALFactory 对象工厂调用 IDAL 接口，访问 SQLServer 数据库的对象 SQLServerDAL 实现了 IDAL 接口。IDAL 接口将两个层次隔开，数据访问层对逻辑层是封闭的。数据的保存是由实体类对象 Model 来完成的。实现数据操作的动作由 SQLServerDAL 类提供的方法完成。

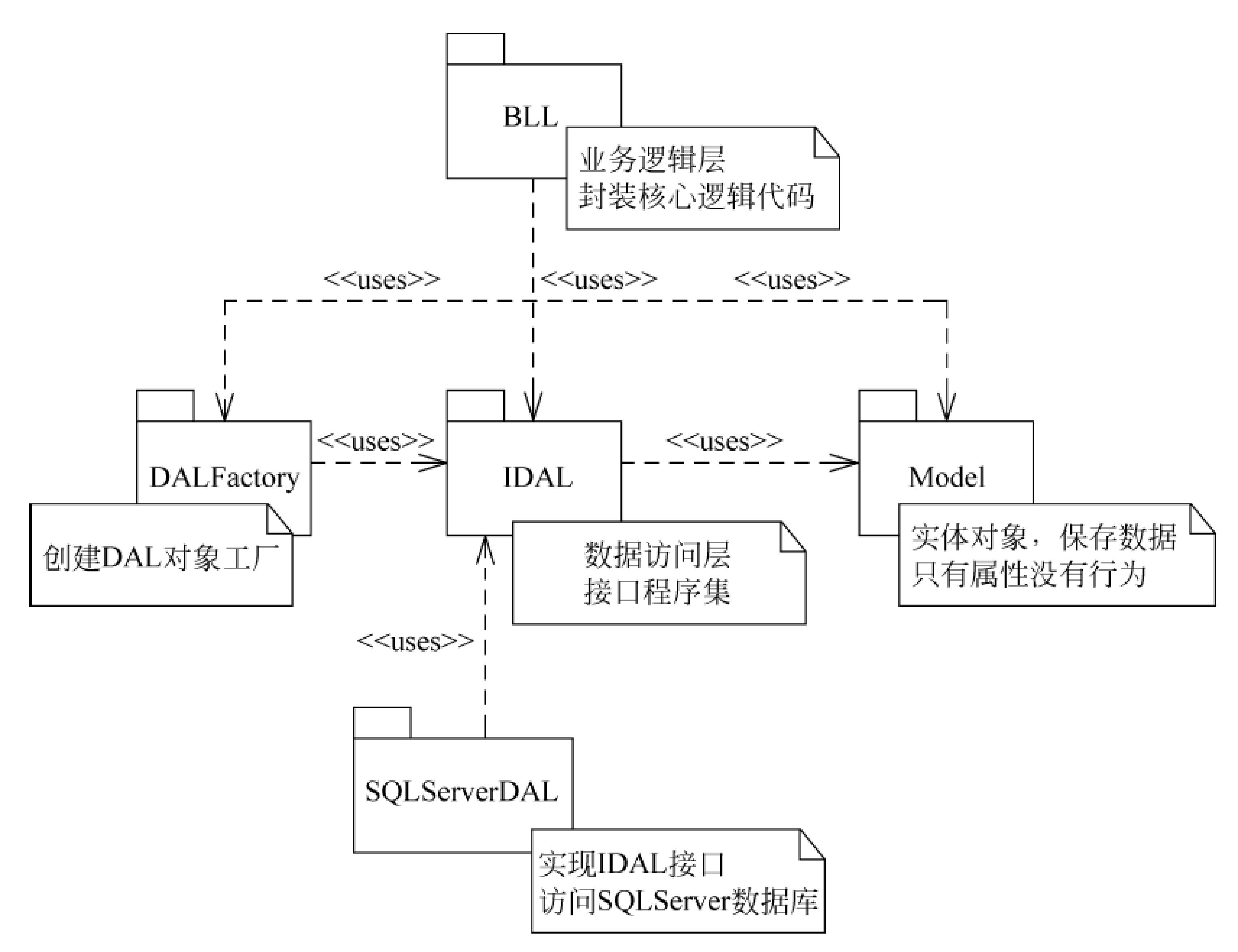


图 8-11　逻辑层和数据访问层的模块结构图

下面的步骤将实现各层次间的处理类。

(1) 设计封装订单信息的实体类

购物流程中，从用户界面录入的配送地址、购物清单和最终形成的订单可以使用实体对象进行保存。实体对象不具有持久化功能，只作为数据的载体，便于业务逻辑针对相应的数

据表进行读写操作。表示层、逻辑层、数据访问层之间的订单交互可以将实体对象作为参数进行交互。数据访问层从数据库读取的数据也可以保存在实体对象中。购物流程需要用到3个实体对象：保存配送地址信息实体类 AddressInfo，保存图书信息的实体类 OrderItemInfo，保存订单信息的实体类 OrderInfo。

设计配送地址实体类，保存用户的配送地址信息。代码如下：

```
public class AddressInfo{
    //定义内部成员变量,保存用户姓名、地址、邮编、电话和 E-mail
    private string realName;
    private string address;
    private string zipCode;
    private string phone;
    private string email;

    // 默认构造函数
    public AddressInfo() { }

    // 重载构造函数,接受外部输入的地址信息
    public AddressInfo(string realName, string address, string zipCode, string
            phone, string email){
        this.realName = realName;
        this.address = address;
        this.zipCode = zipCode;
        this.phone = phone;
        this.email = email;
    }

    // 姓名属性
    public string RealName{
        get { return realName; }
    }
    //详细地址属性
    public string Address{
        get { return address; }
    }
    //邮编属性
    public string ZipCode{
        get { return zipCode; }
    }
    //电话属性
    public string Phone{
        get { return phone; }
    }
    //E-mail 属性
    public string Email{
        get { return email; }
    }
}
```

设计封装购物清单的实体类，保存购买图书的信息。代码如下：

```
public class OrderItemInfo{
    // 定义内部成员变量,保存图书编号、书名、订购数量和订购价格
    private int bookID;
    private string bookName;
    private int quantity;
    private decimal unitPrice;

    //接受外部输入的图书信息参数
    public OrderItemInfo(int bookID, string bookName, int quantity, decimal
            unitPrice) {
        this.bookID = bookID;
        this.bookName = bookName;
        this.quantity = quantity;
        this.unitPrice = unitPrice;
    }

    // 定义图书编号属性
    public int BookID{
        get { return bookID; }
    }
    //定义书名属性
    public string BookName{
        get { return bookName; }
    }
    //定义购买数量属性
    public int Quantity{
        get { return quantity; }
    }
    //定义购买价格属性
    public decimal BookPrice{
        get { return unitPrice; }
    }
    //定义每本图书的小计价格
    public decimal SubTotalPrice{
        get { return unitPrice * quantity; }
    }
}
```

设计封装订单信息的实体类。代码如下：

```
public class OrderInfo{
    private string userName;
    private DateTime orderDate;
    private DateTime completeDate;
    private AddressInfo shippingAddress;
    private OrderItemInfo[] orderItems;

    public OrderInfo(){}

    public OrderInfo(string userName, DateTime orderDate, DateTime
            completeDate, AddressInfo shippingAddress, OrderItemInfo[]
```

```
            orderItems) {
        this.userName = userName;
        this.orderDate = orderDate;
        this.completeDate = completeDate;
        this.shippingAddress = shippingAddress;
        this.orderItems = orderItems;
    }

    // 定义用户名属性
    public string UserName{
        get { return userName; }
        set { userName = value; }
    }

    //定义订购日期属性
    public DateTime OrderDate{
        get { return orderDate; }
        set { orderDate = value; }
    }

    //定义订单处理日期属性
    public DateTime CompleteDate{
        get { return completeDate; }
        set { completeDate = value; }
    }

    //定义订购地址属性,该属性数据类型为地址实体类
    public AddressInfo ShippingAddress{
        get { return shippingAddress; }
        set { shippingAddress = value; }
    }
    //定义购物清单属性,该属性是数据类型为购物清单类的一维数组
    public OrderItemInfo[] OrderItems{
        get { return orderItems; }
        set { orderItems = value; }
    }

    //定义计算订单总价的属性,订单总价为购物清单商品总价
    public decimal OrderTotalPrice{
        get {
            decimal totalPrice = 0;
            if (orderItems!= null) {
                foreach (OrderItemInfo item in orderItems) {
                    totalPrice += item.SubTotalPrice;
                }
            }
            return totalPrice;
        }
    }
}
```

（2）设计订单操作接口

数据库的基本操作包括 Select、Insert、Update 和 Delete，这些操作逻辑仅具有行为而与数据无关，因此可以被抽象为单独的接口模块。购物流程中对订单数据表的操作接口可以定义为 IOrderInfo，接口中定义 3 个操作：添加订单、获取订单和更新订单状态。代码如下：

```
public interface IOrderInfo{
    //添加订单操作
    Int32 InsertOrder(OrderInfo order);
    //获取订单操作
    OrderInfo GetOrder(Int32 orderId);
    //更新订单状态操作
    int UpdateOrderState(Int32 orderID);
}
```

将数据实体和数据库操作分离，符合面向对象的精神，使得两者之间的依赖减弱，当数据行为发生改变时，并不影响 Model 模块中的数据实体对象，避免了因一个类职责过多、过大，从而导致该类的引用者发生“灾难性”影响。

（3）实现订单操作接口

本步骤使用类 SqlOrderInfo 实现订单操作的接口，完成与数据库的交互。类 SqlOrderInfo 继承接口 IOrderInfo，包含三个公共成员函数和两个私有公共函数，代码如下：

```
public class SqlOrderInfo:IOrderInfo
{
      //将带参数的 SQL 语句和参数变量定义成内部私有字符串常量,
      //这些常量在成员函数中使用.
      //添加订单基本信息的 SQL 语句: SQL_INSERT_ORDER ;
      //添加订单商品信息的 SQL 语句: SQL_INSERT_ITEM ;
      //查询订单详细信息的 SQL 语句: SQL_SELECT_ORDER;
      //更新订单状态的 SQL 语句: SQL_UPDATE_STATE;

      //此处定义参数字符串常量.

      //获取订单参数的内部私有成员函数
      private static SqlParameter[] GetOrderParameters(){ }

      //获取订单项参数的内部私有成员函数
      private static SqlParameter[] GetItemParameters(int i){ }

      //添加订单的公共成员函数,添加成功返回订单 ID
      public int InsertOrder(OrderInfo order) { }

      //查询订单的公共成员函数,参数为订单 ID
      public OrderInfo GetOrder(Int32 orderId) { }

      //更新订单的状态,参数为订单 ID
      public int UpdateOrderState(Int32 orderID) { }
}
```

下面对类 SqlOrderInfo 进行详细的分析。

① 定义参数字符串常量。SQL 语句中使用到的参数变量需要定义，代码如下：

```
//用户登录名参数
private const string PARM_USERNAME = "@UserName";
//用户真实姓名参数
private const string PARM_REALNAME = "@RealName";
//订购日期参数
private const string PARM_DATE = "@OrderDate";
//送货详细地址参数
private const string PARM_ADDRESS = "@ShipAddress";
//邮编参数
private const string PARM_ZIPCODE = "@ZipCode";
//联系电话参数
private const string PARM_PHONE = "@Phone";
//E-mail 参数
private const string PARM_EMAIL = "@Email";
//订单编号参数
private const string PARM_ORDERID = "@OrderId";
//图书编号参数
private const string PARM_BOOKID = "@BookID";
//图书名参数
private const string PARM_BOOKNAME = "@BookName";
//订购数量参数
private const string PARM_QUANTITY = "@Quantity";
//销售价参数
private const string PARM_PRICE = "@UnitPrice";
```

② 获取订单参数的成员函数 GetOrderParameters。成员函数 GetOrderParameters 的功能是缓存或读取操作订单基本信息 SQL 语句的参数，将 SQL 语句常量 SQL_INSERT_ORDER 作为键值，存储该 SQL 语句对应的参数数组。使用时，首先使用 SqlHelper 类的 GetCachedParameters 方法读取 SQL_INSERT_ORDER 键值对应的参数数组，如果读取结果不为空，则返回 SqlParameter 类型参数数组，否则，创建 SqlParameter 参数数组变量 parms，并将订单基本信息参数存储到参数数组中，然后返回数组变量 parms。代码如下：

```
private static SqlParameter[] GetOrderParameters(){
    SqlParameter[] parms = SqlHelper.GetCachedParameters(SQL_INSERT_ORDER);
    if (parms == null) {
        parms = new SqlParameter[] {
              new SqlParameter(PARM_USERNAME, SqlDbType.NVarChar, 20),
              new SqlParameter(PARM_DATE, SqlDbType.DateTime, 8),
              new SqlParameter(PARM_REALNAME, SqlDbType.NVarChar,50),
              new SqlParameter(PARM_ADDRESS, SqlDbType.VarChar, 50),
              new SqlParameter(PARM_ZIPCODE , SqlDbType.VarChar, 6),
              new SqlParameter(PARM_PHONE, SqlDbType.NVarChar, 50),
              new SqlParameter(PARM_EMAIL, SqlDbType.NVarChar, 100)};
        SqlHelper.CacheParameters(SQL_INSERT_ORDER, parms);
    }
    return parms;
}
```

③ 获取订单项参数的成员函数 GetItemParameters。成员函数 GetItemParameters 的作用是缓存或读取操作订单中购物清单 SQL 语句所需参数变量，如果缓存为空，则创建参数变量保存到缓存中。参数 i 为订单中商品的序号。代码如下：

```
private static SqlParameter[] GetItemParameters(int i) {
        SqlParameter[] parms =
                SqlHelper.GetCachedParameters(SQL_INSERT_ITEM + i);
        if (parms == null) {
                parms = new SqlParameter[] {
                        new SqlParameter(PARM_BOOKID + i,
                SqlDbType.Int, 4),
                        new SqlParameter(PARM_BOOKNAME + i,
                SqlDbType.NVarChar, 50),
                        new SqlParameter(PARM_QUANTITY + i,
                SqlDbType.Int, 4),
                        new SqlParameter(PARM_PRICE + i,
                SqlDbType.Decimal, 8)};
                SqlHelper.CacheParameters(SQL_INSERT_ITEM + i, parms);
        }
        return parms;
}
```

④ 产生订单的成员函数。定义产生订单的成员函数 InsertOrder，将客户的订单添加到数据库中，参数 order 为订单对象。函数的返回值为 int 型，添加成功返回订单号，否则返回 0。首先要构造添加订单基本信息的 SQL 语句和添加商品清单信息的 SQL 语句。

添加订单基本信息的 SQL 语句：

```
private const string SQL_INSERT_ORDER = "Declare @ID int; Declare @ERR int; INSERT INTO
t_order (UserName, OrderDate, RealName, Address, ZipCode, Phone, Email) VALUES (@UserName,
@OrderDate,@RealName, @ShipAddress,@ZipCode,@Phone,@Email); SELECT @ID = @@IDENTITY;
SELECT @ERR = @@ERROR;";
```

上述语句中首先声明两个变量@ID 和@ERR，用于保存产生的订单 ID 和出错数，当没有错误时，@ERR 应为 0。然后使用 INSERT 语句向订单表 t_order 中添加订单基本信息，@@IDENTITY 返回新增订单号，@@ERROR 返回出错信息数目。

添加订单商品清单信息的 SQL 语句：

```
private const string SQL_INSERT_ITEM = "INSERT INTO t_orderdetail (OrderID,BookID,BookName,
Quantity,UnitPrice) VALUES ( ";
```

上述 SQL 语句向订单详细表 t_orderdetail 中添加商品清单信息，参数列表在 InsertOrder 成员函数中添加。

成员函数中使用 StringBuilder 类对象保存产生订单的 SQL 语句。成员函数定义如下：

```
public int InsertOrder(OrderInfo order) {
    //订单变量 orderID 初值为 0
    int orderID = 0;
    //创建 StringBuilder 类对象 strSQL
```

```
    StringBuilder strSQL = new StringBuilder();

    // 获取订单参数数组
    SqlParameter[] orderParms = GetOrderParameters();

    SqlCommand cmd = new SqlCommand();

    // 为参数变量赋值
    orderParms[0].Value = order.UserName;
    orderParms[1].Value = order.OrderDate;
    orderParms[2].Value = order.ShippingAddress.RealName;
    orderParms[3].Value = order.ShippingAddress.Address;
    orderParms[4].Value = order.ShippingAddress.ZipCode;
    orderParms[5].Value = order.ShippingAddress.Phone;
    orderParms[6].Value = order.ShippingAddress.Email;

    foreach (SqlParameter parm in orderParms)
        cmd.Parameters.Add(parm);

    // 创建连接数据库的对象,执行 SQL 语句,将订单保存到数据库
    using (SqlConnection conn = new SqlConnection(SqlHelper.ConnectionStringLocalTransaction)) {
        // 将 SQL 字符串 SQL_INSERT_ORDER 添加到 strSQL 对象中
        strSQL.Append(SQL_INSERT_ORDER);
        SqlParameter[] itemParms;
        //将 SQL_INSERT_ITEM 字符串添加到 strSQL 对象中
        //将参数添加到 strSQL 中构造添加商品清单的 SQL 语句
        int i = 0;
        foreach (OrderItemInfo item in order.OrderItems) {
            strSQL.Append(SQL_INSERT_ITEM).Append(" @ID").Append(", @BookID").Append(i).
Append(", @BookName").Append(i).Append(", @Quantity").Append(i).Append(", @UnitPrice").
Append(i).Append("); SELECT @ERR = @ERR + @@ERROR;");

            //获取参数变量,并赋值
            itemParms = GetItemParameters(i);

            itemParms[0].Value = item.BookID;
            itemParms[1].Value = item.BookName;
            itemParms[2].Value = item.Quantity;
            itemParms[3].Value = item.SubTotalPrice;
            //将参数对象添加到 Parameters 中
            foreach (SqlParameter parm in itemParms)
                cmd.Parameters.Add(parm);
            i++;
        }

        conn.Open();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = strSQL.Append("SELECT @ID, @ERR").ToString();

        // 执行 SQL 语句,SQL 语句执行完毕,关闭连接对象
```

```
            using (SqlDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {
                // 读取变量@ERR 的值
                rdr.Read();
                // 如果错误数大于 0,则抛出异常,否则返回订单号
                if (rdr.GetInt32(1) != 0)
                    throw new ApplicationException("写入订单信息时产生错误");
                else
                    orderID = rdr.GetInt32(0);
            }
            //清除参数
            cmd.Parameters.Clear();
        }
        return orderID;
    }
```

⑤ 查询订单的成员函数。定义查询订单的成员函数 GetOrder,成员函数的返回值为 OrderInfo 对象类型,函数的参数为 orderId,依据订单号查询订单客户订单。查询订单信息的 SQL 语句定义如下:

```
private const string SQL_SELECT_ORDER = "SELECT o.UserName, o.OrderDate, o.RealName,
o.Address, o.ZipCode, o.Phone, o.Email, o.CompleteDate, d.BookID, d.BookName, d.Quantity,
d.UnitPrice FROM t_order o, t_orderdetail d WHERE o.OrderID = d.OrderID AND o.OrderId =
@OrderId";
```

上述 SQL 语句从订单基本信息表 t_order 和购物清单表 t_orderdetail 中读取订单的详细信息。

成员函数定义如下:

```
public OrderInfo GetOrder(Int32 orderId) {
    OrderInfo order = new OrderInfo();
    //创建一个接受订单号的参数
    SqlParameter parm = new SqlParameter(PARM_ORDERID, SqlDbType.Int);
    parm.Value = orderId;

    //执行查询订单的 SQL 语句
    using (SqlDataReader rdr = SqlHelper.ExecuteReader(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, SQL_SELECT_ORDER, parm)) {

        if (rdr.Read()){
          //建立订单中保存发货地址的地址对象,
          //并将订单的地址信息保存在对象中
            AddressInfo shippingAddress = new AddressInfo(rdr.GetString(2), rdr.GetString(3),
rdr.GetString(4), rdr.GetString(5), rdr.GetString(6));

            //重载订单对象的构造函数,将订单信息保存到订单对象中
            DateTime complateDate = DateTime.MinValue;
            if (!rdr.IsDBNull(7)) {
                complateDate = rdr.GetDateTime(7);
            }
             order = new OrderInfo(rdr.GetString(0), rdr.GetDateTime(1), complateDate,
```

```
shippingAddress, null);

            //建立一个可以保存一组订单项的集合
            IList<OrderItemInfo> orderItemsList = new List<OrderItemInfo>();

            //订单项对象初始化为空
            OrderItemInfo item = null;

            //将订单的每个订单项封装到 item 对象中,并依次添加到 orderItemsList 集合中
            do{
                item = new OrderItemInfo(rdr.GetInt32(8), rdr.GetString(9), rdr.GetInt32(10),
rdr.GetDecimal(11));
                orderItemsList.Add(item);
            } while (rdr.Read());

            //声明订单对象 order 中保存购物清单信息的订单项 OrderItems 数组,数组长度为
orderItemsList 集合元素个数
            order.OrderItems = new OrderItemInfo[orderItemsList.Count];

            //将集合中的元素依次复制到数组中
            orderItemsList.CopyTo(order.OrderItems,0);
        }
    }
    //返回订单对象
    return order;
}
```

⑥ 更新订单状态。当网店工作人员进行订单处理时,需要更新订单状态和处理日期,SQL 语句定义如下:

```
private const string SQL_UPDATE_STATE = "UPDATE t_order SET OrderState = '1',CompleteDate =
GETDATE() WHERE OrderID = @OrderID";
```

上述 SQL 语句中将订单基本信息表 t_order 中的订单状态更改为"1",表示订单处理完毕,完成日期取当前系统日期。

更新订单状态的成员函数定义为 UpdateOrderState,参数为订单号。

```
public int UpdateOrderState(Int32 orderID){
    //接受订单号
    SqlParameter parm = new SqlParameter(PARM_ORDERID, SqlDbType.Int);
    parm.Value = orderID;
    //执行更新订单状态的 SQL 语句
    int retValue = 0;
    retValue = SqlHelper.ExecuteNonQuery(SqlHelper.ConnectionStringLocalTransaction,
CommandType.Text, SQL_UPDATE_STATE, parm);
    //返回记录更新成功的行数
    return retValue;
}
```

(4) 设计返回订单操作接口对象的工厂类

设计工厂类 DataAccess,定义工厂类的成员函数 CreateOrderInfo,工厂类负责实例化

接口对象，将业务逻辑层和数据访问层隔离。

```
public sealed class DataAccess {
    //创建订单工厂类对象
    public static IOrderInfo CreateOrderInfo(){
        return (IOrderInfo)(new SqlOrderInfo());
    }
}
```

(5) 设计逻辑层的订单操作类

订单操作的业务逻辑层使用工厂类实现订单添加、订单查询和订单修改操作。

```
public class Order{
    //创建订单操作的工厂对象
    private static readonly IOrderInfo dal = DataAccess.CreateOrderInfo();
    public Order(){}
    //添加订单业务逻辑处理
    public static Int32 InsertOrder(OrderInfo order) {
        return dal.InsertOrder(order);
    }
    //查询订单业务逻辑处理
    public static OrderInfo GetOrder(Int32 orderID) {
        return dal.GetOrder(orderID);
    }
    //更新订单状态业务逻辑处理
    public static int UpdateOrderState(Int32 orderID) {
        return dal.UpdateOrderState(orderID);
    }
}
```

3. 任务完成总结

三层架构技术将用户表示层、业务逻辑层和数据访问层分开，采用面向接口的编程技术，减少各模块之间的耦合性，增加了系统的稳定性。

4. 课堂训练与知识拓展

使用三层架构技术和面向接口的编程技术改写项目 4 图书查询功能和项目 5 图书管理功能。

8.4.4 任务 8-4 实现订单处理和订单查询流程

1. 任务介绍

客户下订单后，网店人员需要对订单进行处理。简单的处理过程一般是当商品发出后，更改订单的状态，并填写处理日期，处理日期是操作订单的系统日期。客户可以随时对自己所下的订单进行查询，包括查询已处理的订单和未处理的订单。本任务将设计订单处理模块和订单查询模块。

2. 任务分析

订单处理模块界面可以包含3个功能区：订单状态信息显示区，显示订单号、用户登录名、订购日期、订单状态等基本信息；订单详细信息显示区，包括客户的送货地址信息和购物清单信息；订单处理区，包含订单处理事件。处理模块只显示未处理的订单，单击订单状态信息显示区的“详细”链接，将在订单详细信息显示区显示该订单的详细信息。“订单处理区”和“详细信息显示区”可以作为一个整体用一个容器控件Panel包含。订单处理模块的界面如图8-12所示。

当前位置：　网上书店 >管理订单 >订单处理

订单号	用户登录名	订购日期	处理日期	订单状态	查看详细
100083	**min**	**2009-5-27 16:50:09**		**订单未处理**	**详细**
100084	min	2009-5-27 16:52:31		订单未处理	详细
100086	min	2009-5-31 12:17:20		订单未处理	详细
100087	min	2009-7-4 10:45:19		订单未处理	详细
100098	netbook	2009-7-27 0:00:46		订单未处理	详细
100100	netbook	2009-7-27 22:25:55		订单未处理	详细
100101	netbook	2009-7-27 22:39:13		订单未处理	详细
100102	netbook	2009-7-27 23:06:42		订单未处理	详细
100104	netbook	2009-7-27 23:43:24		订单未处理	详细
100106	netbook	2009-7-31 0:15:56		订单未处理	详细

处理订单

送货地址：
用户姓名：嘟嘟
配送地址：常州信息职业技术学院
邮编：213164
联系电话：86458375
Email：min_huifen@126.com

购物清单：

图书名称	销售价格	订购数量	价格小计
牧羊少年奇幻之旅	￥20.00	1	￥20.00

总价：　￥20.00

图8-12　订单处理模块的界面

客户查询订单模块只有两个功能区：订单状态信息显示区和订单详细信息显示区。客户只有登录系统后才能使用该功能，并且只能查询自己的订单，包括未处理的订单和已处理的订单。客户查询订单模块界面如图8-13所示。

(1) 网店管理人员处理订单

本步骤中对图8-12中网店管理员处理订单的过程进行详细的分析。

首先用下面的代码注册显示订单详细信息的用户控件。

```
<%@ Register TagPrefix = "NetBookuc"
                TagName = "OrderControl"
                Src = "~/Controls/OrderControl.ascx" %>
```

使用Panel容器控件包含“处理订单”Button按钮控件和“订单详细信息”用户控件。控件Panel的Visible属性值为“false”，默认情况下该控件不显示，控件中包含的子控件也被屏蔽。当单击订单状态信息显示区的“详细”链接时，设置Visible属性值为“true”，显示Panel控件，同时显示控件中的子控件。代码如下：

```
<asp:Panel ID = "orderPanel" runat = "server" Visible = "false">
```

当前位置：　网上书店 >账户信息 >我的订单

订单号	用户登录名	订购日期	处理日期	订单状态	查看详细
100098	netbook	2009-7-27 0:00:46		待处理中	详细
100100	**netbook**	**2009-7-27 22:25:55**		**待处理中**	**详细**
100101	netbook	2009-7-27 22:39:13		待处理中	详细
100102	netbook	2009-7-27 23:06:42		待处理中	详细
100104	netbook	2009-7-27 23:43:24		待处理中	详细
100106	netbook	2009-7-31 0:15:56		待处理中	详细
100088	netbook	2009-7-23 19:50:12	2009-7-24 13:51:04	已处理	详细
100089	netbook	2009-7-23 21:54:07	2009-7-24 13:51:17	已处理	详细
100090	netbook	2009-7-23 22:04:26	2009-8-20 20:20:04	已处理	详细
100091	netbook	2009-7-23 22:11:28	2009-8-20 20:20:08	已处理	详细
100092	netbook	2009-7-23 22:14:51	2009-8-20 20:20:30	已处理	详细
100093	netbook	2009-7-23 22:17:53	2009-8-20 20:20:34	已处理	详细
100094	netbook	2009-7-23 22:19:08	2009-8-20 20:20:32	已处理	详细
100095	netbook	2009-7-23 22:20:01	2009-7-24 13:51:28	已处理	详细
100096	netbook	2009-7-23 22:30:35	2009-8-20 20:20:11	已处理	详细
100097	netbook	2009-7-23 22:55:11	2009-7-24 14:03:10	已处理	详细
100099	netbook	2009-7-27 17:48:16	2009-8-20 20:20:20	已处理	详细
100103	netbook	2009-7-27 23:20:57	2009-8-20 20:20:17	已处理	详细
100105	netbook	2009-7-27 23:49:24	2009-8-20 20:20:14	已处理	详细

送货地址：
用户姓名：闵惠芬
配送地址：常州武进牛塘
邮编：213163
联系电话：86458375
Email：zhang_xiaoyi@126.com

购物清单：

图书名称	销售价格	订购数量	价格小计
牧羊少年奇幻之旅	￥25.00	1	￥25.00

总价：　￥25.00

图 8-13　客户查询订单模块界面

```
        <br />
        <asp:Button ID = "btnOrderHandle"
                    runat = "server"
                    Text = "处理订单"
                    Width = "150px"
                    OnClick = "btnOrderHandle_Click" />
        <!-- 使用用户自定义控件 - 订单确认 -->
        <NetBookuc:OrderControl ID = "OrderControl1"  runat = "server" />
</asp:Panel>
```

控件 Button 的 OnClick 属性定义事件处理函数 btnOrderHandle_Click，当单击该事件时，进行事件处理过程，首先执行业务逻辑处理类 Order 的 UpdateOrderState 成员函数，依据订单号更新订单的状态，更新成功则显示订单的详细信息。代码如下：

```
protected void btnOrderHandle_Click(object sender, EventArgs e) {
    if (orderID > 0) {
        if (Order.UpdateOrderState(orderID) > 0) {
            gdvOrder.DataBind();
            orderPanel.Visible = false;
            gdvOrder.SelectedIndex = -1;
        }
    }
}
```

使用 GridView 控件呈现订单状态信息，当选中一笔订单时执行属性

OnSelectedIndexChanged 所定义的事件处理程序 gdvOrder_SelectedIndexChanged。当呈现订单状态信息时执行属性 OnRowDataBound 所定义的数据绑定事件 gdvOrder_RowDataBound。列 Columns 包含 5 个 BoundField 控件和 1 个 CommandField 控件,子控件 BoundField 定义订单状态信息字段,子控件 CommandField 定义"选择"按钮,用于显示订单的详细信息。代码如下:

```
<asp:GridView ID = "gdvOrder"
              runat = "server"
              AutoGenerateColumns = "False"
              DataKeyNames = "OrderID"
              DataSourceID = "orderSqlDataSource"
              OnSelectedIndexChanged = "gdvOrder_SelectedIndexChanged"
              OnRowDataBound = "gdvOrder_RowDataBound">
    <Columns>
        <asp:BoundField DataField = "OrderID" HeaderText = "订单号" InsertVisible = "False"
ReadOnly = "True"
            SortExpression = "OrderID" />
        <asp:BoundField DataField = "UserName" HeaderText = "用户登录名" SortExpression =
"UserName" />
        <asp:BoundField DataField = "OrderDate" HeaderText = "订购日期" SortExpression =
"OrderDate" />
        <asp:BoundField DataField = "CompleteDate" HeaderText = "处理日期" SortExpression =
"CompleteDate" />
        <asp:BoundField DataField = "OrderState" HeaderText = "订单状态" SortExpression =
"OrderState" />
        <asp:CommandField HeaderText = "查看详细" SelectText = "详细" ShowSelectButton =
"True" />
    </Columns>
</asp:GridView>
```

使用 SqlDataSource 控件为 GridView 控件提供数据源。属性 SelectCommand 定义带参的 SQL 语句,查询订单的状态信息,SQL 语句中参数为 OrderState,表示订单的状态标识。子控件 Parameter 对参数进行声明,默认值为"0"表示未处理的订单。控件代码如下:

```
<asp:SqlDataSource ID = "orderSqlDataSource"
                   runat = "server"
                   ConnectionString = "<% $ ConnectionStrings:LocalSqlServer %>"
                   SelectCommand = "SELECT [OrderID], [UserName], [OrderDate], [CompleteDate],
[OrderState] FROM [t_order] WHERE ([OrderState] = @OrderState)"
                   ProviderName = "System.Data.SqlClient">
      <SelectParameters>
          <asp:Parameter DefaultValue = "0" Name = "OrderState" Type = "String" />
      </SelectParameters>
</asp:SqlDataSource>
```

事件 gdvOrder_SelectedIndexChanged 处理函数中使用 Order 类的 GetOrder 方法获取订单信息,订单号作为参数。事件 gdvOrder_SelectedIndexChanged 的处理过程用下面的代码描述。

```
protected void gdvOrder_SelectedIndexChanged(object sender, EventArgs e) {
    try {
        //依据订单号获取订单信息
        orderID = Int32.Parse(gdvOrder.SelectedDataKey["OrderID"].ToString());
        OrderInfo order = Order.GetOrder(orderID);

        //绑定订单信息到自定义控件(订单显示界面)
        OrderControl1.Address = order.ShippingAddress;
        OrderControl1.TotalPrice = order.OrderTotalPrice.ToString("c");
        OrderControl1.OrderDetails = order.OrderItems;
        //显示包含订单详细信息的 Panel 容器控件
        orderPanel.Visible = true;
    }
    catch {
        //屏蔽包含订单详细信息的 Panel 容器控件
    orderPanel.Visible = false;
    }
}
```

事件 gdvOrder_RowDataBound 处理过程对显示的状态标识符进行处理。代码如下：

```
protected void gdvOrder_RowDataBound(object sender, GridViewRowEventArgs e)
    {
    //判读是否是数据行
    if (e.Row.RowType == DataControlRowType.DataRow) {
          //获取订单状态
          string orderState = e.Row.Cells[4].Text;
          //状态标识符为"0",描述为"未处理"
          //状态标识符为"1",的用红色字体描述为"已处理"
          //其余情况设为空字符
          if (orderState == "0") {
               e.Row.Cells[4].Text = "订单未处理";
          }
          else if (orderState == "1") {
               e.Row.Cells[4].Text = "<font color='red'>已处理</font>";
          }
          else {
               e.Row.Cells[4].Text = string.Empty;
          }
    }
}
```

(2) 客户查询订单

客户查询订单模块的设计与步骤(1)中网店人员处理订单的设计过程基本一致。下面仅对不同的部分进行分析。本模块中没有订单的处理事件,此外仅供登录客户查询订单,因此提取订单状态信息的 SQL 语句需包含登录用户名参数。子控件 ProfileParameter 定义的参数 UserName 来源于 Profile 对象。代码如下：

```
<asp:SqlDataSource ID="orderSqlDataSource"
                   runat="server"
```

```
                ConnectionString = "<% $ ConnectionStrings:LocalSqlServer %>"
                    SelectCommand = " SELECT [ OrderID ], [ UserName ], [ OrderDate ],
[CompleteDate], [OrderState] FROM [t_order] WHERE ([UserName] = @UserName)">
    <SelectParameters>
        <asp:ProfileParameter Name = "UserName" PropertyName = "UserName" Type = "String" />
    </SelectParameters>
</asp:SqlDataSource>
```

3. 任务完成总结

订单处理模块和客户查询订单模块的关键点是使用数据源控件获取订单状态信息，并使用 GridView 控件呈现订单的状态信息。当选择一笔订单时，依据订单号从数据库获取订单信息，并使用用户控件呈现订单详细信息。两者具有不同点：处理模块是网店管理人员使用的功能，显示所有未处理的订单，网店管理人员处理订单，需要更改订单的状态标识和处理时间；客户查询订单模块是客户使用的功能，客户可以查询自己的订单，但不可以对订单进行处理。

4. 课堂训练与知识拓展

网店业务管理员可以查询所有已处理的客户订单，编写查询已处理所有客户订单的订单查询模块，该功能模块供网店管理人员使用。已处理的订单数量比较多，在查询时应增加时间区间作为筛选条件。

8.5 项目总结

客户订单管理功能的核心模块是购物流程，一个简单的完整购物流程包括了选购商品形成购物清单、填写配送地址形成订单和将订单存储到数据库，第一步可以做成购物车，最后一步生成购物清单，整个购物流程使用向导控件 Wizard 完成。客户完成购物后，网店管理人员可以对订单进行处理，客户可以查询订单的状态。项目设计的关键技术是多层架构技术和面向接口的编程技术，将用户界面层、逻辑层和数据访问层分开，使用接口进行隔离，减少层与层模块之间的耦合性，增加了系统的稳定性。

8.6 项目实训

1. 任务描述

具有会员信息的网站一般提供用户注册为网站会员的功能。用户注册的过程可以分为几个步骤完成，网站首先要求用户输入主要的登录信息，然后继续输入详细信息，最后一步将用户信息写入数据库。用户注册信息写入数据库前，需要对新用户登录名进行验证，当用户登录名没有被使用时，写入成功，否则需要更换用户登录名。编写网站会员管理模块，实现会员注册功能和查询功能。

2. 任务要求

（1）功能要求

设计会员数据库；实现用户注册功能；实现网站管理员查询注册用户信息功能；实现用户登录网站查询自己信息功能。

（2）技术要求

使用 Wizard 控件设计注册流程；软件架构使用三层架构技术；采用面向接口的编程技术将逻辑层与数据访问层隔开。

项目9

网上书店报表设计

9.1 项目介绍

报表是将业务数据以图表的形式体现,易于直观理解。在程序开发工程中,经常要合并计算、多级汇总、制作图表、条件格式化,这些用普通的数据控件很难完成,利用水晶报表可以简化这些工作。水晶报表可以制作非常漂亮的图表、格式化文本,并可以将报表导出成Word、Excel、PDF、HTML 等格式。VS 2005 集成了水晶报表功能。本项目中将使用水晶报表功能设计网上书店的订单清单报表和图书分类汇总销售统计报表。

9.2 项目分析

本项目中网上书店报表设计主要完成以下几个任务:分别使用 Pull 模式(拉模式)和 Push 模式(推模式)设计销售订单报表,设计订单基本信息和订单明细主从报表、设计图书分类汇总销售统计图表。显示报表需要的数据来源于数据集,报表获取数据集数据有两种方式,Pull 模式和 Push 模式。在 Pull 模式下水晶报表文件根据指定的驱动程序连接数据库,直接从数据库拉数据组装后绑定到报表查询控件显示,该方法实现比较简单。在 Push 模式下程序执行时通过代码连接数据库,从数据库获取数据组装成数据集并送推数据到报表文件,该方式实现报表比较灵活,可以依据报表显示的需要在数据集中对数据重新组织。订单基本信息清单列表和每张订单所对应的商品信息明细表构成主从关系报表,用水晶报表的分组功能可以设计具有主从关系的报表,订单基本信息和订单明细主从关系报表通过订单号关联,从订单主表中选择订单号能够连连接到订单明细表。图书销售统计主要统计每类图书的销售量,统计的数据来源于销售订单,销售量统计主要统计已处理的销售订单,未处理的销售订单不在统计范围中,进行统计报表设计时,按图书类别分类汇总,图书的类别数据来源于图书分类库。

本项目中使用到的图书详细信息表(t_book)和图书分类表(t_category)的结构在项目 4 中已经介绍,订单基本信息表(t_order)和订单明细表(t_orderdetail)的结构在项目 8 中已经介绍。

9.3 相关知识

Crystal Reports 自 1993 年开始就已经是 Visual Studio 的一部分，并且现在已成为 Visual Studio 2005 中的标准报表创建工具。每套 Visual Studio 2005 都附带了该工具，并且它直接集成到开发环境中。它提供了非常丰富的模型以使我们能够在运行时操作属性和方法。

VS.NET 水晶报表主要有以下优点：

① 快速的报表开发。

② 能够导出成为复杂的交互性图表。

③ 可以与其他控件一起在 WebForm 中使用。

④ 能够动态地将报表导出成为.pdf、.doc、.xls、.html、.rtf 等多种格式文档。

基于 B/S 结构的水晶报表的 Web 应用由客户端和服务器端两部分构成。客户端仅需要一个可以访问嵌入 aspx 页面报表的浏览器就可以了。服务器端需要安装水晶报表引擎(Crystal Report Engine (CREngine.dll))，通过它可以完成一些任务，如在报告文件中合并数据，转换报告为其他格式等。也正是因为报告引擎的作用，才可以将 ASP.NET 水晶报表转换成为普通 HTML 格式页面。

ASP.NET 水晶报表在水晶报表设计器(Crystal Report Designer (CRDesigner.dll))中创建，在设计器中可以设计标题、插入数据、公式、图表、子报表等。

水晶报表文件以.rpt 作为文件扩展名。执行报表中的第一步就是在水晶报表设计器接口创建此报表，在默认安装中微软已经提供了一些现成的.rpt 例子。

水晶报表文件连接数据库的方式一般有两种：一种是不需要编写代码，由水晶报表自己选择数据库；另一种是编写代码手动组装数据集 DataSet，然后将数据传送到表报文件。

可以通过水晶报表查看器来呈现水晶报表。水晶报表查看控件是一个 WebForm 控件，可以将它看成是一个在.aspx 页面中存放报表的容器。在一些复杂的操作中，报表服务器与 Web 服务器可能不在同一物理主机上，Web 服务器将 HTTP 请求传送到报表服务器上去。水晶报表也可以当作 WebService 执行。

Crystal Reports 通过数据库驱动程序与数据库连接。每个驱动程序都被编写为可处理特定数据库类型或数据库访问技术。为了向开发人员提供最灵活的数据访问方法，Crystal Reports 数据库驱动程序被设计为可提供数据访问的 Pull 模式和 Push 模式。

(1) Pull 模式

报表被请求时，水晶报表直接根据指定的驱动连接数据库，然后组装数据。在拉模式中，驱动程序将连接到数据库并根据需要将数据“拉”进来。使用这种模式时，与数据库的连接和为了获取数据而执行的 SQL 命令都同时由 Crystal Reports 本身处理，不需要开发人员编写代码。如果在运行时无须编写任何特殊代码，则使用拉模式。

(2) Push 模式

推模式需要开发人员编写代码连接到数据库，执行 SQL 命令创建与报表中的字段匹配的记录集或数据集，并且将该对象传递给报表。该方法可以从 Web.Config 配置文件中获取连接共享，并在 Crystal Reports 收到数据之前先将数据筛选出来。通过编写代码连接

数据库并组装 DataSet,将数据集传送至报表。在这种情况下,通过使用连接共享以及限制记录集合的大小,可以使报表性能最大化。

水晶报表设计器能够直接将报表添加到工程,也能够使用独立的报表对象。报表类型有类型化报表(Strongly-typed)和非类型化报表(Un-Typed)。

(1) Strongly-typed 报表

当报表文件加入到项目中去时,它就变成了一个了 strongly-typed 报表。在这些情况下,将拥有直接创建报表的对象的权力,这将减少一些代码并且能够提供一些性能。

(2) Un-Typed 报表

这里的报表是独立的报表对象,并不直接包含在项目中,因此称为 un-typed 报表。在这种情况下,可以使用水晶报表的 ReportDocuemt 对象建立一个实例,并且手动地调用报表。

9.4 项目实施

下面通过 4 个任务来介绍水晶报表的使用。

9.4.1 任务 9-1 使用水晶报表 Pull 模式显示订单列表

1. 任务介绍

水晶报表的 Pull 模式通过直接连接数据库获得报表所需要的数据。首先用水晶报表设计器设计报表文件,在报表文件中添加直接连接数据库的数据集。其次使用报表查看器关联报表文件,报表查看器嵌入到 aspx 页面中。最后以 Web 页面方式向用户展现报表。本任务将讨论水晶报表 Pull 模式的实现过程。

2. 任务分析

Pull 模式的使用主要分为两个步骤:创建直接连接数据库的水晶报表文件和使用水晶报表查看器呈现水晶报表。使用水晶报表设计器的数据库专家打开“OLE DB(ADO)数据连接”对话框,在对话框中连接数据库,连接数据库时需要配置访问数据库的账户和密码。在代码隐藏类中创建报表对象,将报表文件加载到报表对象中,并实现报表对象与报表查看器的绑定。数据表将使用项目 8 中的 t_order 订单基本表。

(1) 创建使用数据库的水晶报表文件

在 BookShop 解决方案中添加水晶报表文件,首先单击鼠标右键打开“添加新项”对话框,如图 9-1 所示。选择“Crystal 报表”图标,在“名称”一栏的文本框中输入“CrystalReportPull. rpt”文件名,“语言”选择“Visual C#”,单击“添加”按钮将打开“Crustal Reports 库”对话框,如图 9-2 所示。

在图 9-2 中选择“作为空白报表”选项,单击“确定”按钮进入水晶报表设计器界面,如图 9-3 所示。水晶报表设计器的左侧是“字段资源管理器”,可以对水晶报表运用“公式字段”、“运行总计字段”、“特殊字段”等。水晶报表设计器右侧是报表设计区,报表设计区由





图 9-1　添加水晶报表文件



图 9-2　“Crustal Reports 库”对话框

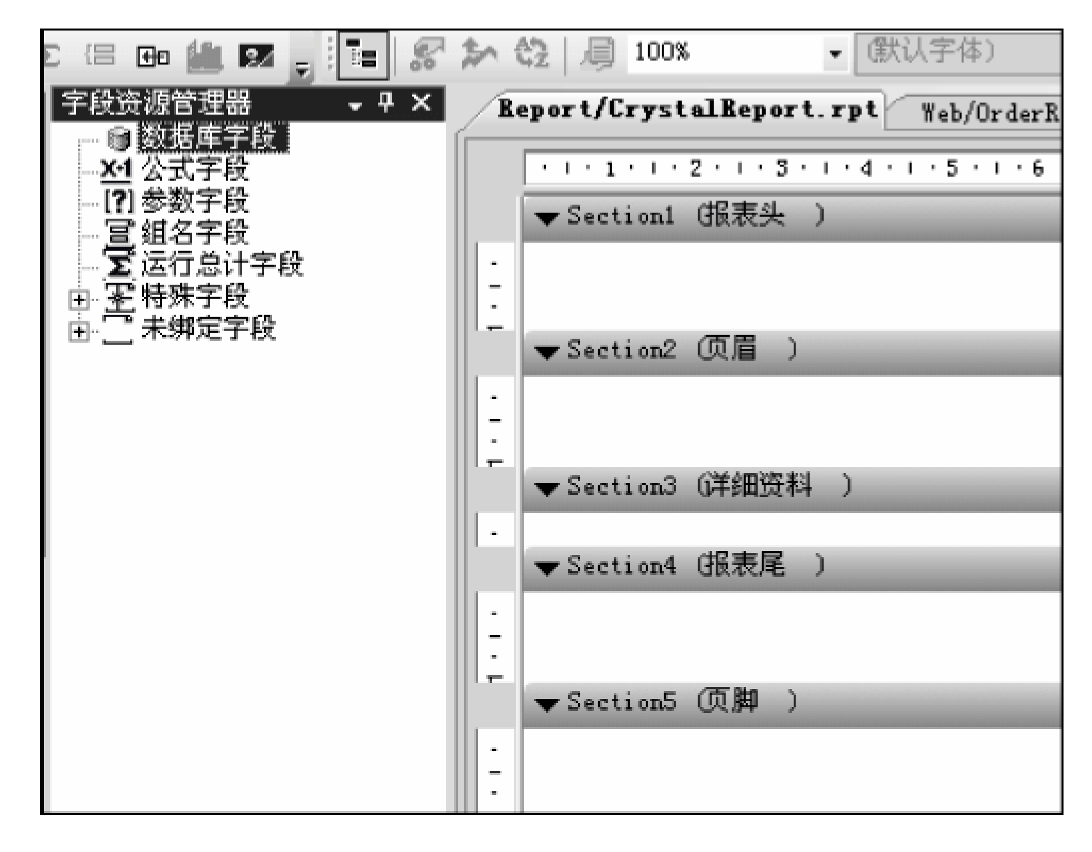


图 9-3　水晶报表设计器界面

5 个部分组成。报表头，可以为整个报表设计一个统一的表头。页眉，为报表要显示的每列数据设置一个列名。详细资料，要显示的报表每列数据的详细内容，一般该列是数据库表的记录。页脚，当分页显示时，共有多少页，当前是第几页等与当前页相关的信息一般在页脚设计。报表尾，统计与汇总信息设计在报表尾。

在水晶报表设计器的字段资源管理器中打开数据库字段的快捷菜单(在“数据库字段”选项上右击)，选择快捷菜单的“数据库专家”打开数据库专家的配置界面，在配置界面中可以进行直接访问数据库的配置。如图 9-4 和图 9-5 所示。

在数据库专家配置界面中展开“创建新连接”，再展开“OLE DB(ADO)”，打开“OLE DB(ADO)”对话框，如图 9-6 所示。选择对话框中提供程序“Microsoft OLE DB Provider for SQL Server”，单击“下一步”按钮将显示连接数据库的连接信息，如图 9-7 所示。连接信息界面提示输入服务器、用户 ID、密码、数据库。依提示输入信息单击“下一步”按钮进入“高



图 9-4 数据库专家配置



图 9-5 创建 SQLServer 数据库连接



图 9-6 “OLE DB(ADO)”对话框

级信息”设置界面如图 9-8 所示。,单击“完成”按钮,数据库 NetBook 将添加到数据库专家界面,如图 9-9 所示。

OLE DB (ADO)
连接信息
提供登录到选定的数据源所需的信息。
服务器(S): .
用户 ID: sa
密码(P): *********
数据库(D): NetBook
集成安全(T):
< 上一步(B)　下一步(N) >　完成　取消

图 9-7　填写连接信息



图 9-8　编辑属性值



图 9-9　添加数据库表

将 NetBook 数据库中的订单表 t_order 筛选到数据库专家对话框的右侧栏“选定的表”区域，单击“确定”按钮，完成了在水晶报表设计器中添加数据库表的操作。图 9-10 中的“数据库字段”选项下面将会出现选中的表 t_order，展开表将显示该表所有的字段。

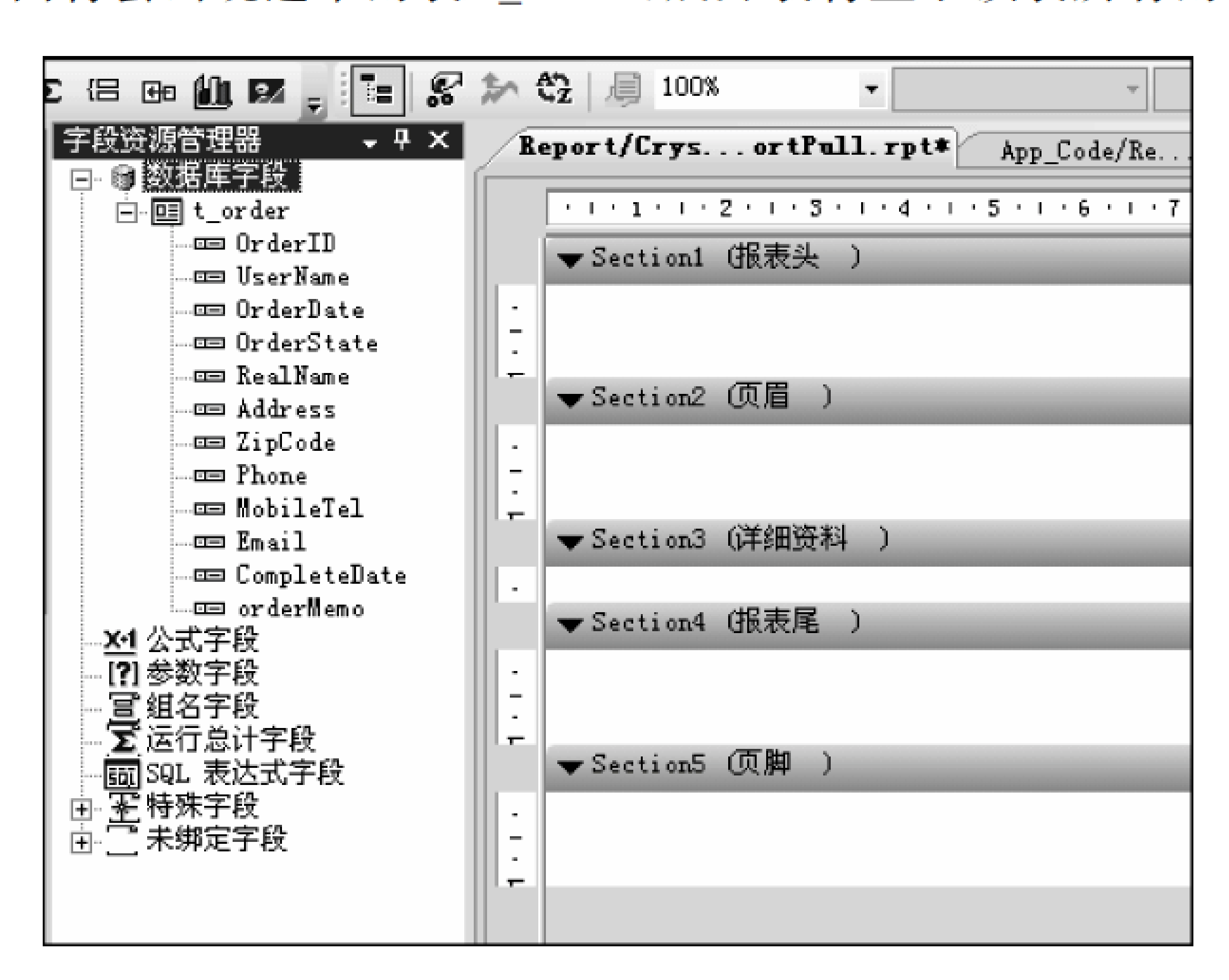


图 9-10　添加订单表的水晶报表设计器

将字段 OrderID(订单编号)、RealName(姓名)、Address(配送地址)、ZipCode(邮编)、Phone(联系电话)和 OrderDate(订购日期)拖放到设计区域的 Section3(详细资料)区域，并将页眉区的英文标题改为中文标题。设计结果如图 9-11 所示。

报表一般会设置一个统一的表头，下面的步骤将在 Session1(报表头)为报表添加表头。在报表头区域右击打开快捷菜单，在快捷菜单中选择“插入”，然后选择二级菜单中的“文本对象”。文本对话框将添加到鼠标停留的地方，在对话框中输入“客户订单”。为使报表头美



图 9-11 添加报表字段到设计界面



图 9-12 插入文本对象

观一些，可以使用“文本对象”的快捷菜单打开“格式编辑器”，对文本字体、大小、颜色等进行设置。如图 9-12 和图 9-13 所示。

使用“黑体”、“常规”样式、大小设为 18、颜色设为“黑色”设置的报表头如图 9-14 所示。

(2) 使用水晶报表查看器查看 Pull 模式订单报表

使用水晶报表查看器可以将报表文件嵌入到 aspx 页面。在解决方案中添加页面文件 OrderReport.aspx。选择工具箱的菜单“报表”下的控件 CrystalReportViewer 拖放到页面，打开 OrderReport.aspx.cs 后台代码文件。

在代码隐含类中首先添加如下的两行命名空间：

```
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
```

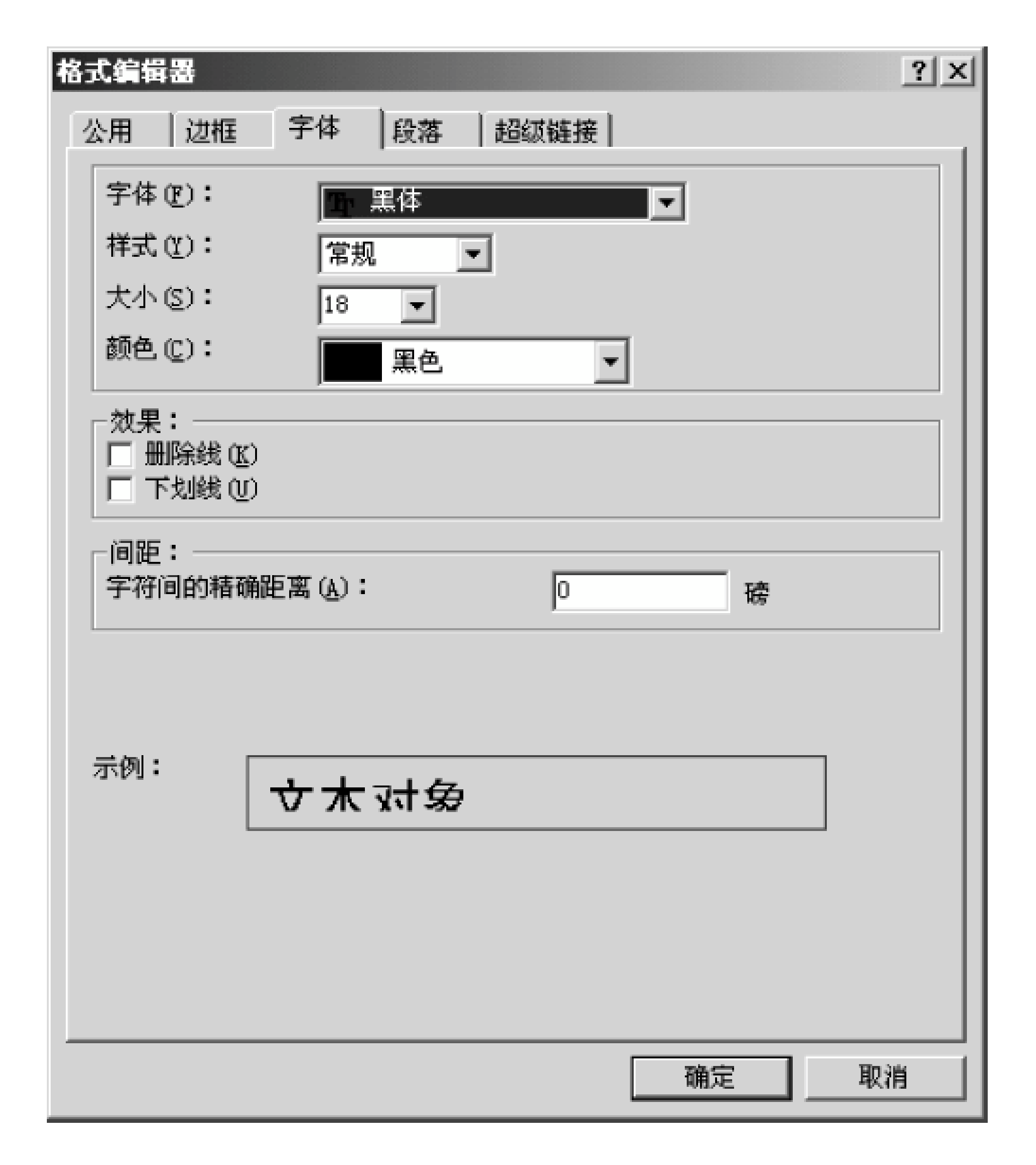


图 9-13　文本格式编辑器

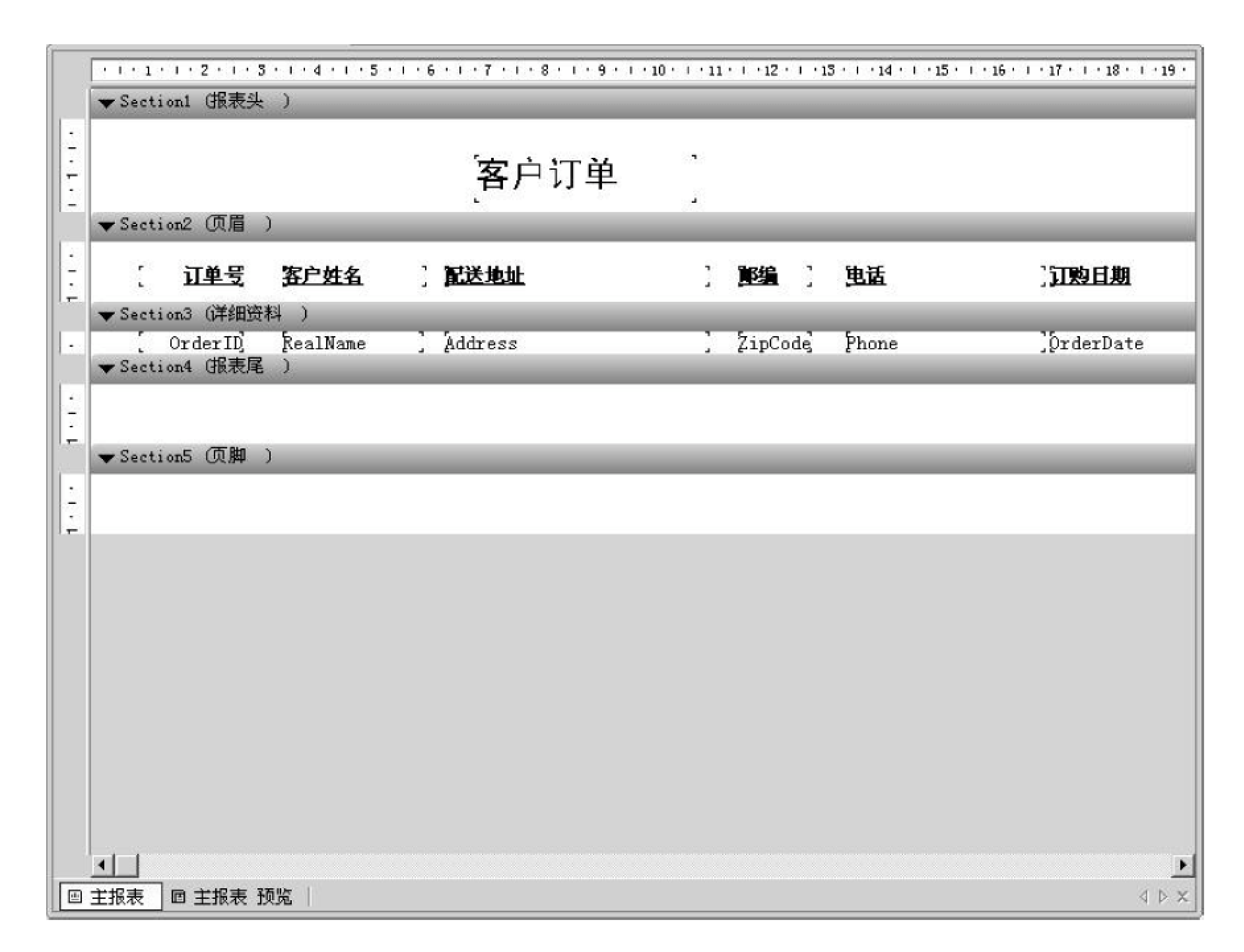


图 9-14　报表布局结果

然后定义 ReportDocument 对象变量用于加载报表文件。

```
ReportDocument reportDoc = new ReportDocument();
```

最后在 Page_Load 函数中添加加载报表的代码,并将报表对象绑定到水晶报表查看器中显示。报表文件访问数据库,需要连接数据库的信息,可以使用 TableLogOnInfo 类实现。

```
protected void Page_Load(object sender, EventArgs e)
{
        reportDoc = new ReportDocument();
        reportDoc.Load(Server.MapPath("~/Report/CrystalReportPull.rpt"));

        TableLogOnInfo logOnInfo = new TableLogOnInfo();
        foreach(CrystalDecisions.CrystalReports.Engine.Table tb in reportDoc.Database.Tables){
                logOnInfo = tb.LogOnInfo;
                logOnInfo.ConnectionInfo.ServerName = ".";
                logOnInfo.ConnectionInfo.DatabaseName = "NetBook";
                logOnInfo.ConnectionInfo.UserID = "sa";
                logOnInfo.ConnectionInfo.Password = "sqlserver";
                tb.ApplyLogOnInfo(logOnInfo);
        }
        CrystalReportViewer1.ReportSource = reportDoc;
        CrystalReportViewer1.DisplayGroupTree = false;
}
```

程序运行后的结果如图 9-15 所示。运行结果除支持 Web 方式显示外,也支持将报表导出成 PDF、Excel 等多种格式的文档。

当前位置: 网上书店 >销售统计 >销售清单(Pull)

1 / 1 主报表 100% Business Objects

客户订单

订单号	客户姓名	配送地址	邮编	电话	订购日期	
100072	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-18	11:21:1
100074	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-25	13:56:4
100075	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-25	14:11:4
100076	闵惠芬1	常州武进牛塘k;	213163	86392594	2009-5-25	15:08:1
100077	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-25	16:16:1
100078	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-25	16:52:1
100079	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-25	16:55:5
100080	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-27	16:29:5
100081	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-27	16:35:3
100082	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-27	16:37:1
100083	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-27	16:50:0
100084	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-27	16:52:3
100085	嘟嘟	常州信息职业技术学院	213164	86458375	2003-2-21	0:41:0
100086	嘟嘟	常州信息职业技术学院	213164	86458375	2009-5-31	12:17:2
100087	嘟嘟	常州信息职业技术学院	213164	86458375	2009-7-4	10:45:19
100088	闵惠芬	常州武进大学城信息学院	213163	86458375	2009-7-23	19:50:1
100089	闵惠芬	常州武进大学城信息学院	213163	86458375	2009-7-23	21:54:0
100090	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:04:2
100091	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:11:2
100092	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:14:5
100093	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:17:5
100094	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:19:0
100095	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:20:0
100096	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:30:3
100097	闵惠芬	常州武进牛塘	213163	86458375	2009-7-23	22:55:1
100098	闵惠芬	常州武进牛塘	213163	86458375	2009-7-27	0:00:4
100099	闵惠芬	常州武进牛塘	213163	86458375	2009-7-27	17:48:1
100100	闵惠芬	常州武进牛塘	213163	86458375	2009-7-27	22:25:5
100101	闵惠芬	常州武进牛塘	213163	86458375	2009-7-27	22:39:1

图 9-15 订单运行结果

3. 任务完成总结

使用 Pull 模式访问水晶报表，是使用 OLE DB(ADO)数据库连接驱动将数据库表直接添加到水晶报表文件中，页面呈现水晶报表时，通过水晶报表查看器访问水晶报表文件，水晶报表文件直接从数据库中"拉"数据，加载到报表文档对象中，并将报表文档对象绑定到查看器上显示。需要注意的是水晶报表文件访问数据库的连接驱动需配置账户和密码，同时代码加载结果集也需要配置访问数据库的账户和密码。

4. 课堂训练与知识拓展

项目 4 中给出网上书店数据库中记录图书详细信息的表 t_book。使用 Pull 模式设计报表文件显示 t_book 表中字段 bookid(图书 ID)、bookname(图书书名、publisher(图书出版社)、pubdate(出版时间)和 price(图书价格)信息。

9.4.2 任务 9-2 使用水晶报表 Push 模式设计客户订单报表

1. 任务介绍

任务 9-1 中讨论了 Pull(拉)模式水晶报表的使用，下面讨论 Push(推)模式水晶报表的使用。Push 模式首先要创建一个强类型 DataSet 数据集，然后创建报表文件，并将报表文件指定给数据集。在代码中访问数据库，并将结果存储到数据集中，由数据集推送给报表文件。这种用代码推送数据的方式与 Pull 直接使用数据库连接驱动访问数据库的方式是有区别的，一个是主动获取数据；一个是被动获取数据。需要注意的是在 Push 模式中编程组装的 Dataset 里的 SQL 语句中的字段要与水晶报表里的 SQL 语句字段一致。简单地说，Push 模式中的水晶报表是个模板，把在设计器里报表的格式设计好后，再组装 DataSet 就可以生成报表了。

本任务中将使用 Push 模式建立客户订单报表。

2. 任务分析

先建立 DataSet 数据架构文件 OrderDataSet.xsd，以后将用它生成强类型数据集，数据架构文件中包含订单字段 OrderID(订单编号)、RealName(姓名)、Address(配送地址)、ZipCode(邮编)、Phone(联系电话)和 OrderDate(订购日期)。创建报表文件 CrystalReportPush.rpt，将报表文件指定给架构文件生成的强类型数据集。在 aspx 页面上拖放一个水晶报表查看器控件 CrystalReportViewer，同时将其与报表文件关联，在代码中访问数据库并把数据存入 DataSet。请求报表文件时，数据集将数据推送到报表文件。通过以上步骤实现水晶报表的 Push 模式。

(1) 创建报表文件数据集

创建 Push 模式的数据集要使用解决方案中的添加新项功能。打开"添加新项"对话框，选择"Visual Studio 已安装的模板"中的"数据集"，命名创建包含 DataSet 类的 XML 架构文件"OrderDataSet.xsd"，语言使用"Visual C#"，如图 9-16 所示。单击"添加"按钮，应

用程序将打开数据集设计器。从工具箱中将 TableAdapter 控件拖放到数据集设计界面，打开“TableAdapter 配置向导”对话框的选择数据库连接界面，如图 9-17 所示。



图 9-16　添加数据集



图 9-17　“TableAdapter 配置向导”对话框

选择 Web.config 配置文件中的连接“NetBookConnectionString”，单击“下一步”按钮进入配置向导的“选择命令类型”对话框，如图 9-18 所示。选择第一项“使用 SQL 语句”，单击“下一步”按钮进入图 9-19 所示的输入 SQL 语句界面。

在输入 SQL 语句界面文本输入框中输入访问数据库的 SQL 语句，语句代码如下：

```
SELECT  [OrderID]
       ,[RealName]
       ,[Address]
```

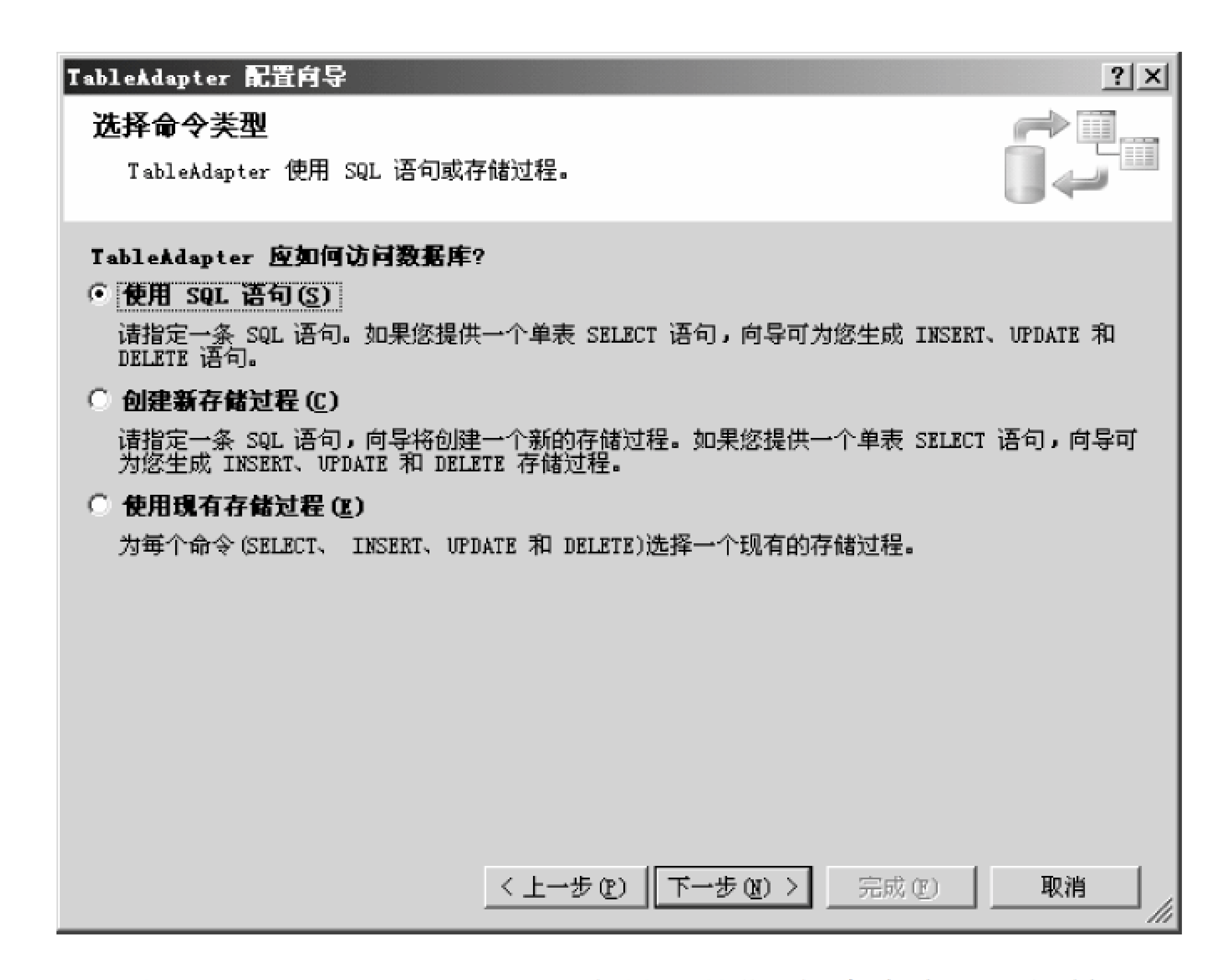


图 9-18 TableAdapter 配置向导下的“选择命令类型”对话框



图 9-19 TableAdapter 配置向导“输入 SQL 语句”

```
        ,[ZipCode]
        ,[Phone]
        ,[OrderDate]
FROM [NetBook].[dbo].[t_order]
```

输入 SQL 查询语句后，在上述界面中单击“下一步”按钮，进入为数据集添加方法的界面，如图 9-20 所示。使用系统默认方法，单击“下一步”按钮进入向导结果界面，如图 9-21 所示，单击“完成”按钮，完成使用向导配置数据集的过程。

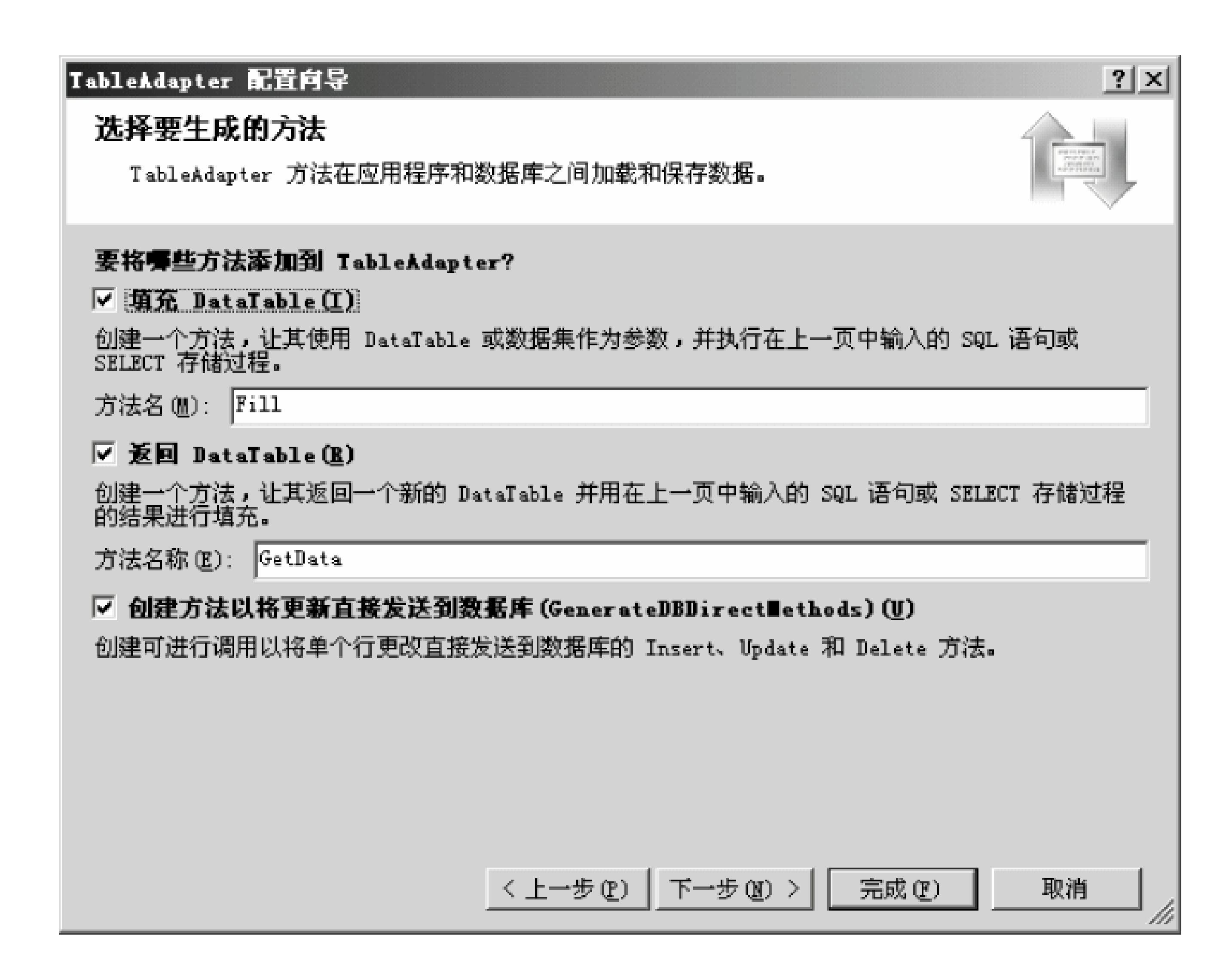


图 9-20　选择要生成的方法

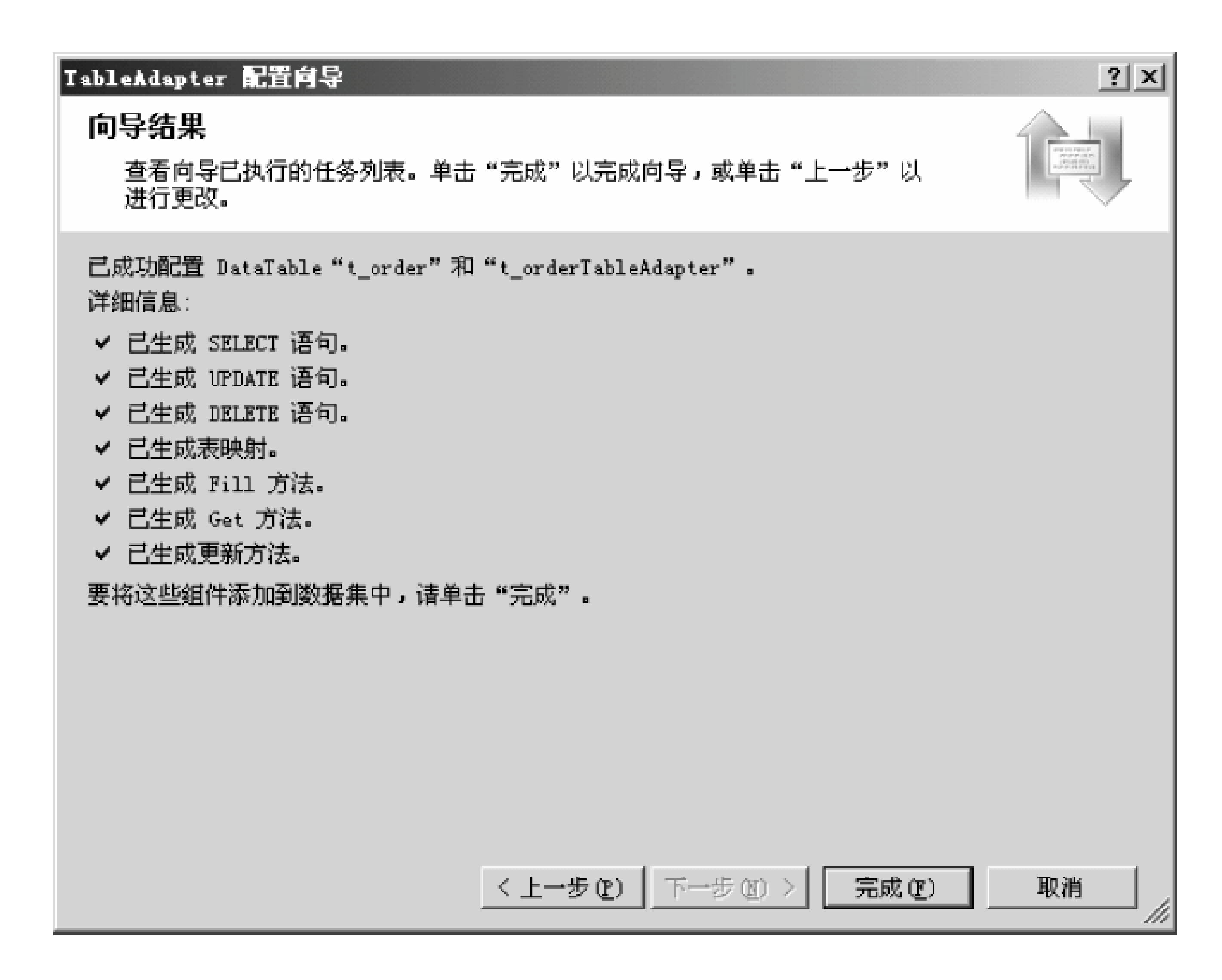


图 9-21　向导结果

完成数据集配置后，将在数据集设计器中产生 OrderDataSet.xsd 数据架构文件。架构文件数据集如图 9-22 所示。

(2) 设计关联数据集的报表文件

使用解决方案的添加新项功能添加一个使用 Push 模式的水晶报表文件，文件命名 CrystalReportPush.rpt，添加过程与 Pull 模式类似。在给水晶报表文件分配数据集时不同于 Pull 模式。Pull 模式是使用“数据库专家”的“OLE DB(ADO)”数据库驱动直接连接数据库。Push 模式是使用“数据库专家”中“项目数据”选项下的“ADO.NET 数据集”实现数据库的访问。

展开“数据库专家”的可用数据库选项“ADO.NET 数据集”，在“ADO.NET 数据集”下显示步骤(1)中创建的数据架构文件生成的数据集 OrderDataSet，将 OrderDataSet 数据集添加到“数据库专家”的右侧“选定的表”文本框，如图 9-23 所示。



图 9-22　订单数据架构

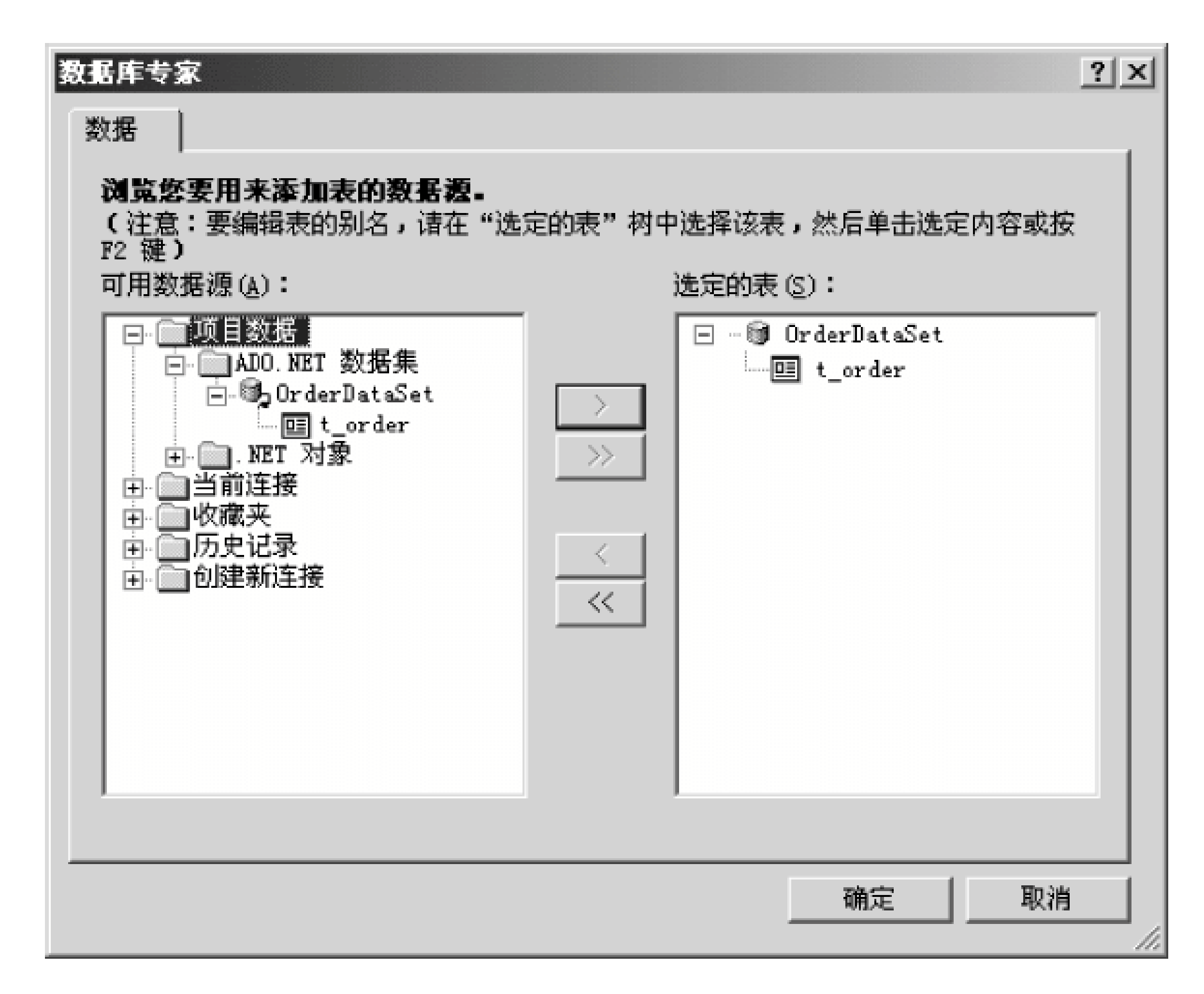


图 9-23　报表文件关联数据集

在上述界面中单击“确定”按钮，将在 Push 模式报表设计器界面的字段资源管理器的数据库字段下建立 t_order 数据表，数据表的字段与 OrderDataSet 数据集中表 t_order 字段对应。表的名称使用数据集中的表名称。设计界面如图 9-24 所示。

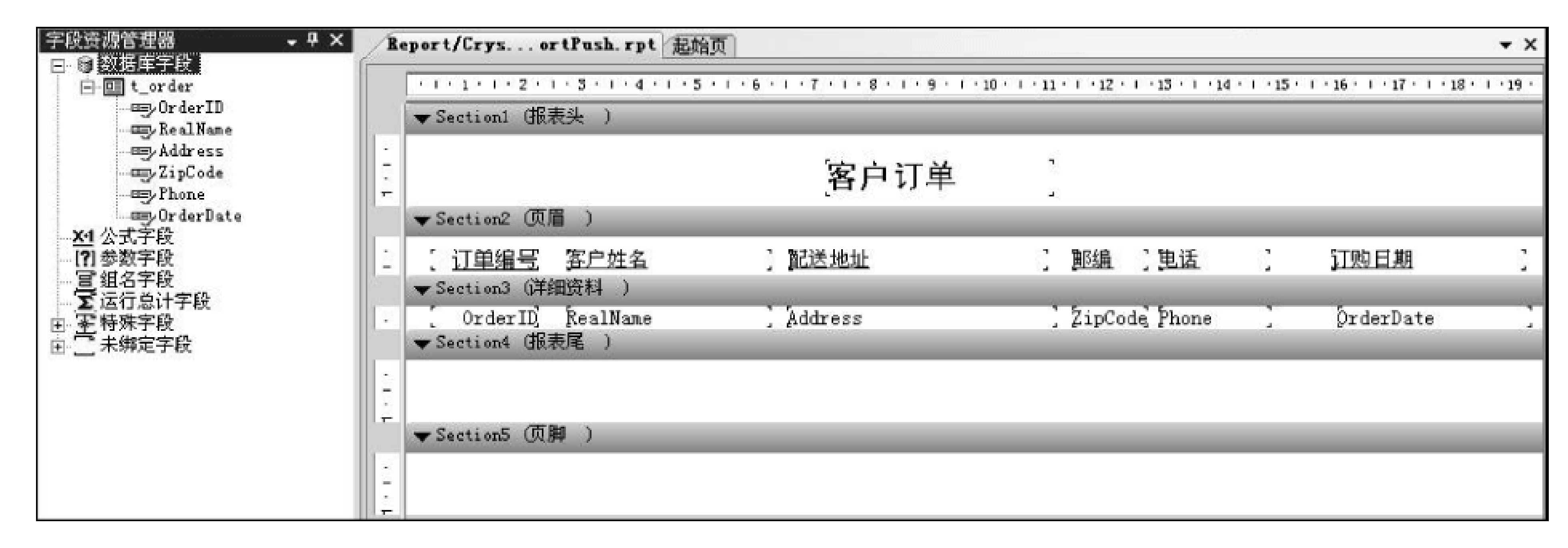


图 9-24　Push 模式报表文件设计器界面

(3) 使用水晶报表查看器查看 Push 模式订单报表

建立 Push 模式水晶报表显示页面文件 OrderPushReport.aspx，从 WebForm 工具栏中拖动水晶报表查看器控件 Crystal Report Viewer 至 OrderPushReport.aspx 页面中。使用 Push 模式水晶报表需要访问数据库，除引用水晶报表命名空间外，还需使用访问 SQLServer 数据库的命名空间 System.Data.SqlClient。在页面隐藏类文件 OrderPushReport.aspx.cs 中添加如下 3 个命名空间：

```
using System.Data.SqlClient;
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
```

在 Page_Load 方法中添加加载数据库数据到数据集的代码，并添加指派数据集到报表文件的代码，代码实现过程如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    //建立水晶报表对象
    ReportDocument reportDoc = new ReportDocument();
    //建立 XSD 数据集对象
    OrderDataSet ords = new OrderDataSet();

    try {
        //加载数据集
         OrderDataSetTableAdapters.t_orderTableAdapter da = new OrderDataSetTableAdapters.t_
orderTableAdapter();
        da.Fill(ords.t_order);

        //加载水晶报表文件
        reportDoc.Load(Server.MapPath("~/Report/CrystalReportPush.rpt"));

        //指派数据集到水晶报表对象
          reportDoc.SetDataSource(ords);
      }
      catch (System.Exception ex) {
        throw new Exception("数据处理错误!", ex);
     }
      //绑定报表文件到水晶报表查看器
      CrystalReportViewer1.ReportSource = reportDoc;
      CrystalReportViewer1.DisplayGroupTree = false;
}
```

在浏览器中浏览页面 OrderPushReport.aspx，可以获得与图 9-15 类似的报表。

3. 任务完成总结

本任务中实现了水晶报表 Push 模式的使用。首先创建 XSD 架构文件，并生成强类型数据集，将要显示的数据库中的订单表 t_order 及字段映射到数据集，然后创建报表文件，将报表文件与数据集关联，最后在 aspx 页面中使用报表查看器呈现水晶报表。代码中需将数据集加载到报表文件中。程序在运行时，数据集从数据库取数据推送到报表中。

4. 课堂训练与知识拓展

使用 Push 模式设计水晶报表显示任务 9-1 中课堂训练与拓展中任务，显示 t_book 表中字段 bookid(图书 ID)、bookname(图书书名)、publisher(图书出版社)、pubdate(出版时间)和 price(图书价格)信息。

9.4.3　任务 9-3　设计订单和订单商品明细主从报表

1. 任务介绍

在报表中，有许多报表是主从表结构，比如订单与订单商品明细，订单基础信息是订单

基础表中的一条记录,订单商品明细信息是订单明细表中多条记录,两张表通过订单编号关联,这种具有主从数据关系的报表可利用水晶报表分组功能实现。本任务将介绍网上订单的订单基础表和订单明细表的主从报表实现。

2. 任务分析

建立主从报表主要从如下几个步骤来进行。首先要建立主从报表数据集,将订单基础表和订单明细表添加到数据集中,可以使用任务 9-2 中的方法先建立主从报表架构文件,然后再生成强类型数据集。

(1) 建立订单主从报表数据集架构

本步骤中将使用服务器资源管理器建立主从报表的数据集 MasterDetailDataset。在解决方案中创建 MasterDetailDataset.xsd 文件,打开数据集设计器界面。从“服务器资源管理器”中选择“数据连接”,如果“数据连接”菜单下没有访问 NetBook 数据库的连接,则使用“数据连接”快捷菜单“添加新连接”功能添加一个连接 NetBook 数据库的连接。打开 NetBook 数据库连接下的表菜单,将 NetBook 数据库“表”下的 t_order 表和 t_orderdetail 表添加到订单主从表设计器界面。通过订单号关联两张表,如图 9-25 所示。

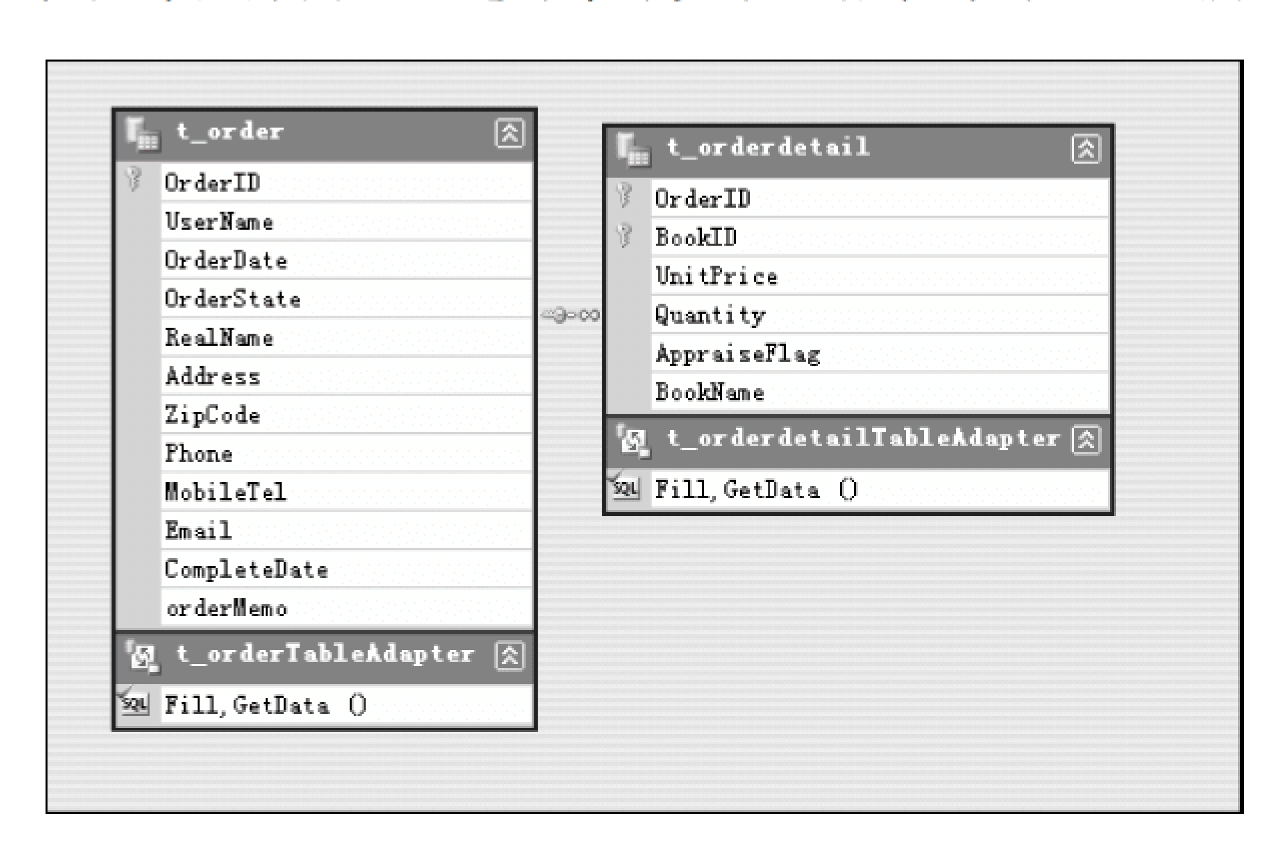


图 9-25 订单主从报表数据架构文件

(2) 建立订单主从报表文件

使用水晶报表数据库专家添加数据集。打开数据库专家对话框,展开可用数据源“项目数据”下“ADO.NET 数据集”,将“MasterDetailDataset”数据集添加到“选定的表”区域,数据集中包含两张具有主从关系的表 t_order 和 t_orderdetail,如图 9-26 所示。单击“链接”选项卡,选择“按关键字”建立链接,将两张表按订单号链接在一起,如图 9-27 所示。

使用报表专家将数据集添加到水晶报表设计器中后,在设计器界面“字段资源管理器”的“数据库字段”下多出两张表“t_order”和“t_ordertail”,在设计器区右击,选择“报表”下的“组专家”。将“可用字段”区域的“t_order.OrderID”选择到右侧“分组依据”中,依据订单号建立分组,如图 9-28 所示。使用任务 9-1 和任务 9-2 中所介绍的方法,拖动字段到设计区域。设计结果如图 9-29 所示。

(3) 使用水晶报表查看器查看主从报表

在解决方案中建立页面文件 MasterDetailOrderReport.aspx,将工具栏中水晶报表查



图 9-26　数据库专家添加主从报表数据集



图 9-27　数据库专家建立主从报表链接



图 9-28　“组专家”对话框

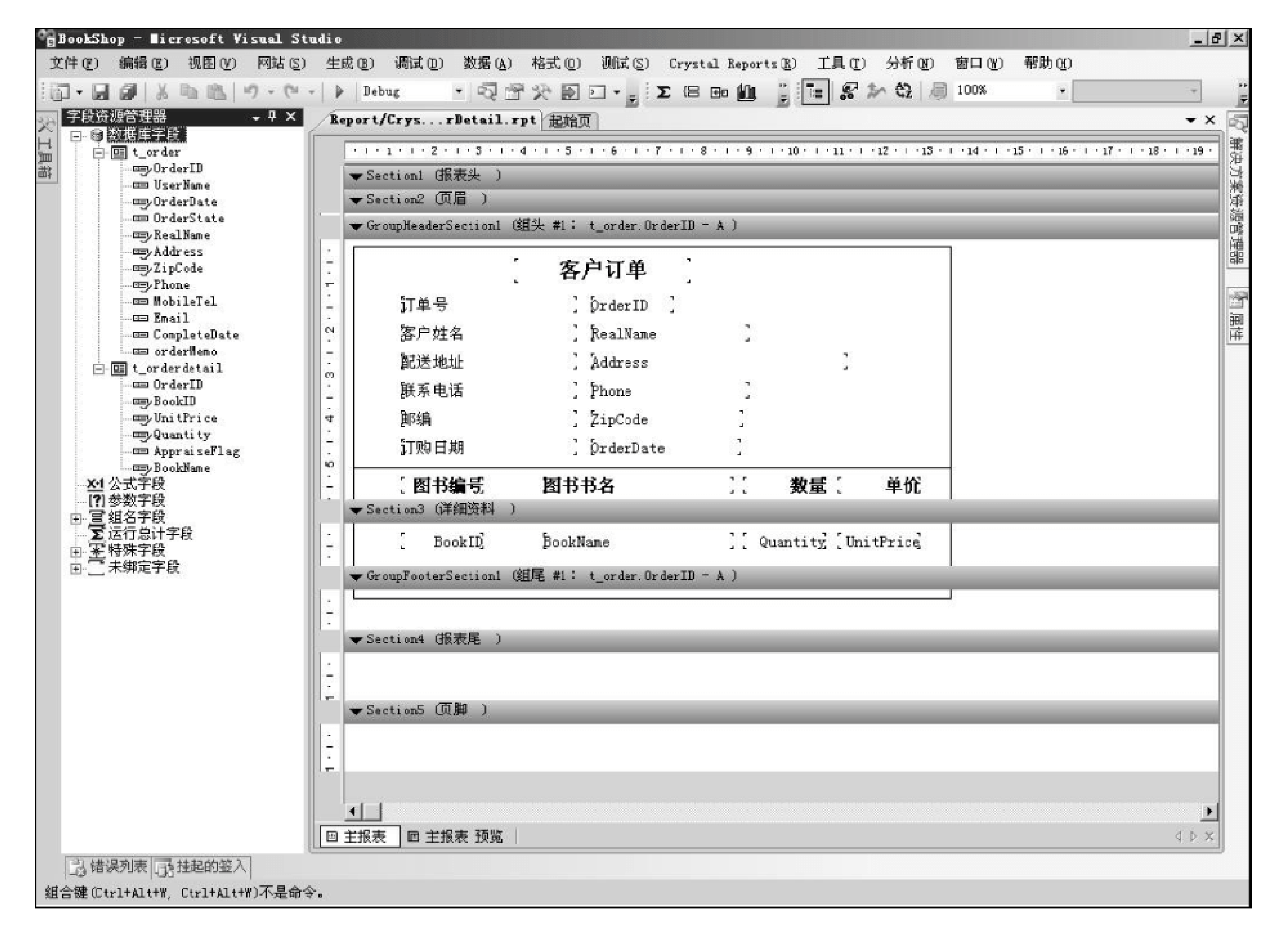


图 9-29　使用主从报表设计器设计报表

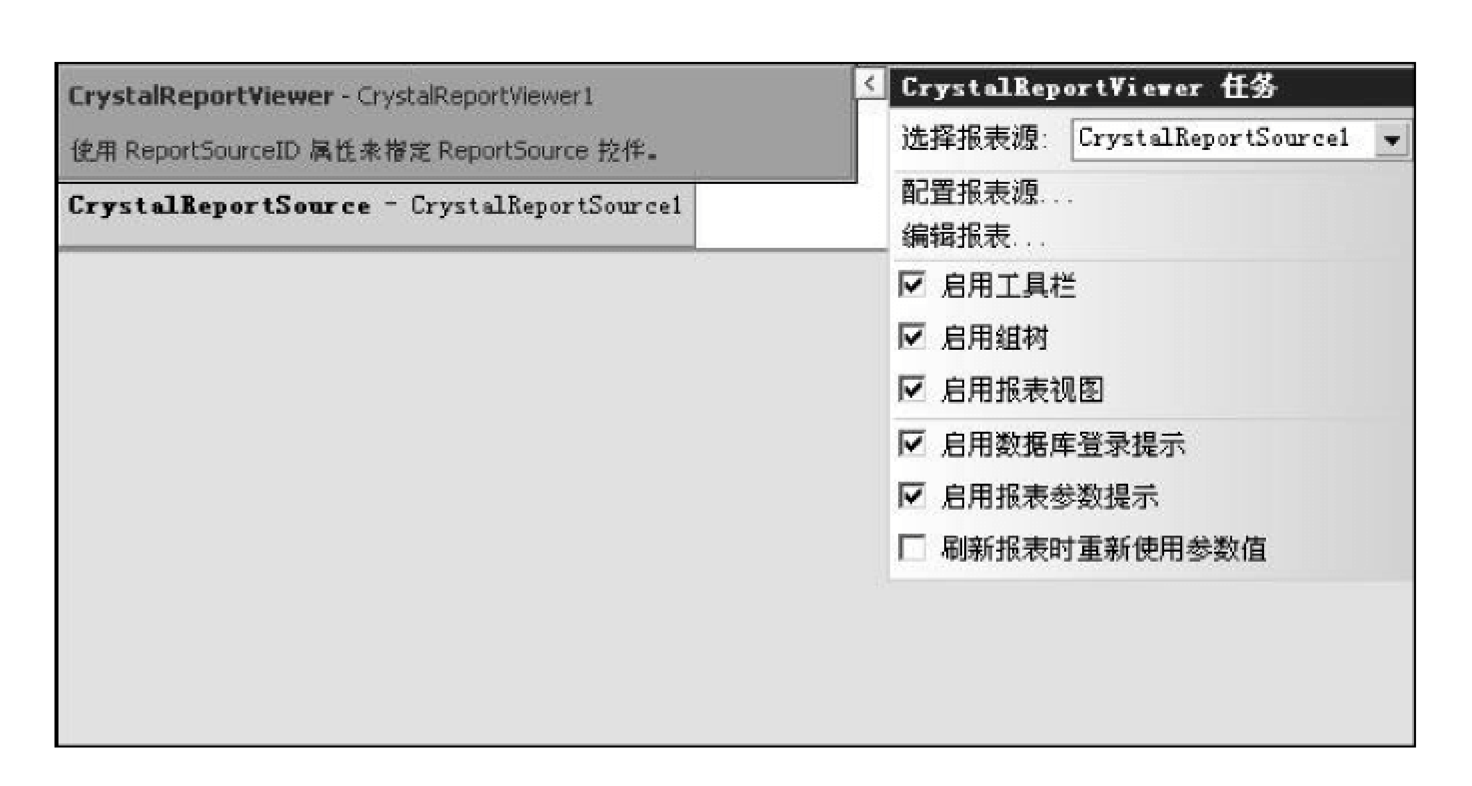


图 9-30　主从报表 aspx 页面设计

看器控件 CrystalReportViewer 和水晶报表数据源控件 CrystalReportSource 添加到页面上，产生 CrystalReportViewer1 和 CrystalReportSource1 两个实例。打开水晶报表查看器控件 CrystalReportViewer1 的智能感应开关，选择报表源 CrystalReportSource1，其他选项选默认值。建立水晶报表查看器和水晶报表数据源的联系，如图 9-30 所示。

下面的代码完成建立强类型数据集，以及向水晶报表推送数据的过程。

```
protected void Page_Load(object sender, EventArgs e) {
    //实例化数据集
    MasterDetailDataSet mdDataSet = new MasterDetailDataSet();
```



```
    //构建数据集与订单明细表的关联
    MasterDetailDataSetTableAdapters.t_orderdetailTableAdapter detailda = new MasterDetailDataSetTableAdapters.t_orderdetailTableAdapter();

    detailda.Fill(mdDataSet.t_orderdetail);

    //构建数据集与订单基础表的关联
    MasterDetailDataSetTableAdapters.t_orderTableAdapter orderda = new MasterDetailDataSetTableAdapters.t_orderTableAdapter();

    orderda.Fill(mdDataSet.t_order);

    // 加载水晶报表文件
    CrystalReportSource1.ReportDocument.Load(Server.MapPath("~/Report/CrystalReportMasterDetail.rpt"));

    //为水晶报表文档对象指定数据集
    CrystalReportSource1.ReportDocument.SetDataSource(mdDataSet);
}
```

代码执行后产生如图 9-31 所示的结果。左侧一列为主报表订单号，右侧一列为每张订单的主辅明细。



图 9-31　主从报表结果 Web 显示

3. 任务完成总结

本任务中讨论了主从报表的设计过程。需要注意的是建立报表数据集时，要将主表和从表添加到报表数据集中，并在数据集中建立主从表的主外键的关系。具体设计报表时需要建立一个分组，依据分组字段建立主从报表连接。

4. 课堂训练与知识拓展

图书分为若干个类别，每一类别下有许多图书，建立图书类别和图书明细之间的主从报表。

9.4.4 任务 9-4 建立图书销售量统计图表

1. 任务介绍

用图表描述统计数据比较直观,一目了然。本任务中将讨论网店图书销售量统计图表的制作方法,讨论使用直方图显示依据图书类别统计图书销售量。

2. 任务分析

使用图表显示统计数据,首先必须建立绘制统计图所需要的数据集,建立数据集过程可以参考本章中的任务 9-2 和任务 9-3 中介绍的方法进行。按图书类别统计图书销售量,要建立包含图书类别的销售订单明细,只有处理完毕的销售订单才能够加入统计过程。建立数据集后,使用报表专家将数据集添加到报表设计器。使用报表设计器的插入图表功能在报表设计器中绘制销售量统计图表。最后建立报表显示页面,在页面上使用报表查看器关联报表文件,使用 Push 模式显示查询结果。

(1) 建立订单销售明细数据集

从订单基础表 t_order 和订单明细表 t_orderdetail 中可以获取订单详细资料。订单基础表 t_order 中的字段 OrderState 描述了订单的状态,当 OrderState 的值为"1"时表示订单已经处理完毕,订单明细中的销售数据可以参与销售量的统计。图书类别信息存储在类别表 t_category 中。图书基本信息表 t_book 中存储了图书类别 ID(categoryID)。销售订单明细表 t_orderdetail 中存储了图书 ID(BookID)。订单基础表 t_order、订单明细表 t_orderdetail、图书基本信息表 t_book 和图书类别表 t_category 通过主外键关联。

建立销售订单明细视图 vw_OrderSaleDetail,视图中包含订单编号 OrderID、图书编号 BookID、书名 BookName、订购数量 Quantity、订购单价 UnitPrice、图书类别 ID categoryID、图书类别名称 categoryName。

创建订单销售明细视图的 SQL 语句如下:

```
create view vw_OrderSaleDetail as
select    d.OrderID
         ,d.BookID
         ,d.BookName
         ,d.Quantity
         ,d.UnitPrice
         ,c.categoryID
         ,c.categoryName
from t_order o,t_orderdetail d,t_book b,t_category c
where b.categoryID = c.categoryID
  and d.BookID = b.bookID
  and o.OrderID = d.OrderID
and o.OrderState = '1'
```

要对图书的销售量进行分类汇总,只需对销售订单明细视图 vw_OrderSaleDetail 中的字段订购数量 Quantity 按图书类别名称 categoryName 进行分组汇总就可以了。获取依据图书类别统计销售量的 SQL 语句如下:

```
select   SUM(Quantity) as SumQuantity
        ,categoryName
from vw_OrderSaleDetail
group by categoryName
```

本任务中执行上述 SQL 语句得到表 9-1 所示的数据：

表 9-1　执行 SQL 语句后结果

类别	教育	少儿	生活	艺术
销售量(本)	6	2	2	31

建立数据架构文件 SaleDataSet. xsd，将视图 vw_OrderSaleDetail 添加到架构文件设计器中，建立包含图书类别的销售订单明细数据集。如图 9-32 所示。

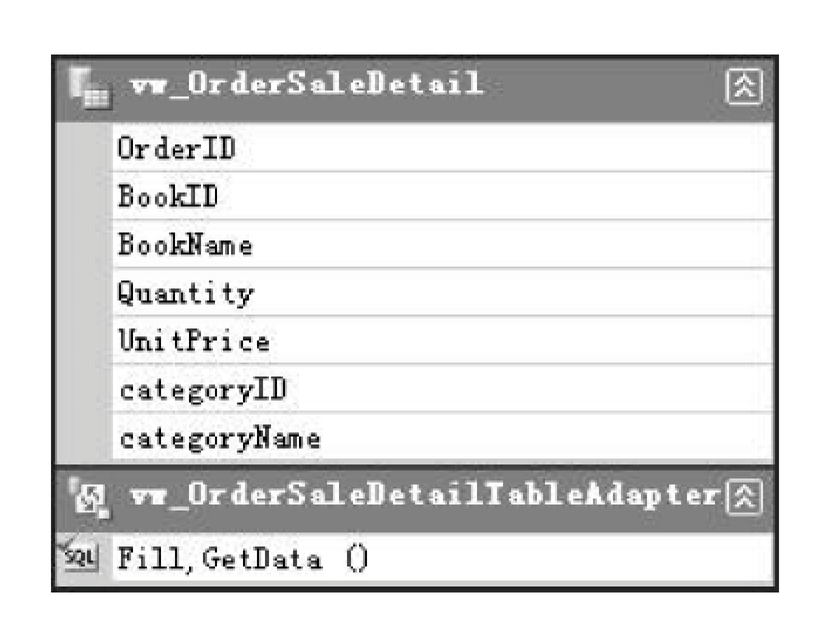


图 9-32　包含图书类别的销售订单明细数据集

(2) 建立销售量分类统计直方图报表文件

建立图书销售量统计直方图报表文件分为两步：一是使用数据库专家将数据集添加到报表设计器；二是在报表设计器中插入图表。在解决方案中添加报表文件 CrystalReportChart. rpt，使用数据库专家添加数据集 SaleDataSet 到报表设计器，如图 9-33 所示。在报表设计器空白区右击，使用快捷菜单插入图表功能添加图标，如图 9-34 所示。



图 9-33　使用数据库专家添加销售订单明细数据集

使用插入图表功能将打开水晶报表的“图表专家”对话框，选择“图表专家”对话框中“类型”选项卡，图表类型区域将显示条形图、折线图、饼图等多种图表类型。本任务中使用直方图描述图书分类统计销售量，因此选择“条形图”作为图表类型，“条形图”布局采用“垂直”布



图 9-34　使用报表设计器插入图表

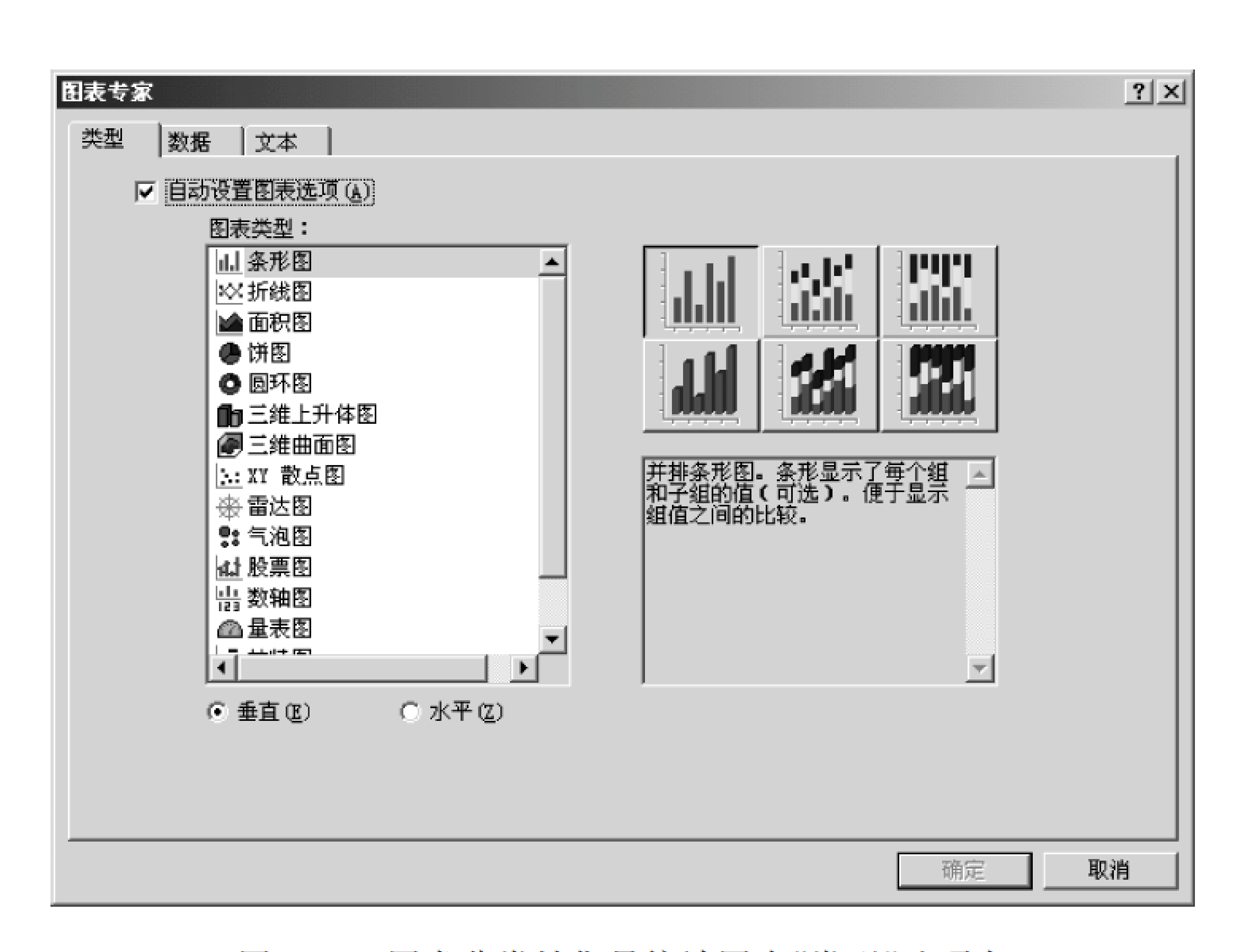


图 9-35　图书分类销售量统计图表“类型”选项卡

局方式，如图 9-35 所示。

设置了图表类型后，需要设置图表数据。图表类型是图表在呈现时，采用的图形化表示方式，呈现图表所需要的参数可以使用图表数据功能选项设置。图 9-36 中“可用字段”文本

选择框中上方显示报表字段，下方显示 ADO.NET 数据集表字段。将数据集表字段当中的图书类别名称字段 categoryName 作为分类统计条件添加到“变更主体”下方文本框中。将订单销售明细数据集中销售量字段 Quantity 作为汇总统计字段添加到“显示值”区域，单击“设置汇总运算”按钮打开图 9-37 所示的“编辑汇总”对话框，选择字段 Quantity 作为汇总字段，在“计算此汇总”栏中选择汇总方式“求和”。做了上述设置之后，报表在运算时将自动按图书类别对图书销售量进行分类求和汇总，并以直方图形式显示。



图 9-36　图书分类销售量统计图表“数据”选项卡

图表文本功能选项卡可以设计统计图表的标题、副标题、脚注、数据标题等内容。图 9-38 显示了图表专家“文本”选项卡的设置项。“标题”分组区的设置项是图表常用的基本设置，去掉“自动添加文字”选项，设置“标题”为“按图书类别统计销售量”，设置“组标题”为“图书类别”，设置数据标题为“图书销售量”，其他内容为空。“格式”分组区可以对各标题选项的字体进行设置。当完成设置后单击“确认”按钮将报表设计器中显示可视化的样表，如图 9-39 所示。



图 9-37　图书分类销售量统计图表“编辑汇总”对话框

在样表上右击，选择“图表选择”，继续选择二级菜单“模板”将打开“选择图表类型”对话框，如图 9-40 所示。在“图库”选项卡中选择“柱状图”，布局使用“成簇”，勾选“使用深度”复选框，经过上述设置后将会获得具有立体感的直方图。单击“确认”按钮完成设置。

(3) 使用报表查看器显示统计结果图

在解决方案中添加页面文件 SaleSumReport.aspx，在页面上添加水晶报表查看器控件 CrystalReportViewer1 和水晶报表数据源控件 CrystalReportSource1。将报表数据源绑定



图 9-38　图书分类销售量统计图表“文本”选项卡

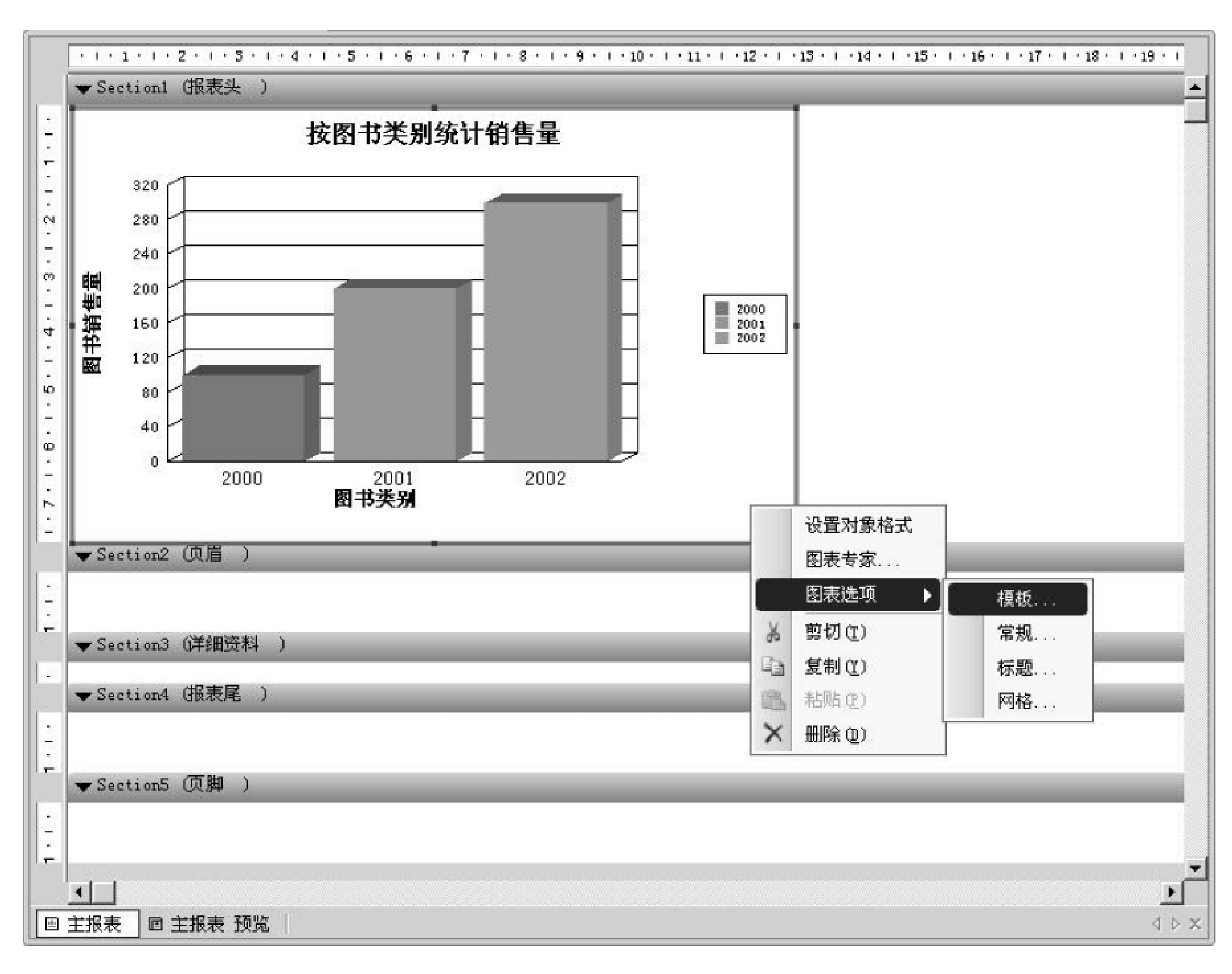


图 9-39　图书分类销售量统计图表样表

到上一步骤中设置的报表文件 CrystalReportChart.rpt。将报表查看器控件 CrystalReportViewer1 绑定到报表数据源控件 CrystalReportSource1。在页面文件对应的隐含类文件 SaleSumReport.aspx.cs 中添加加载数据集到报表文件的代码。代码如下：

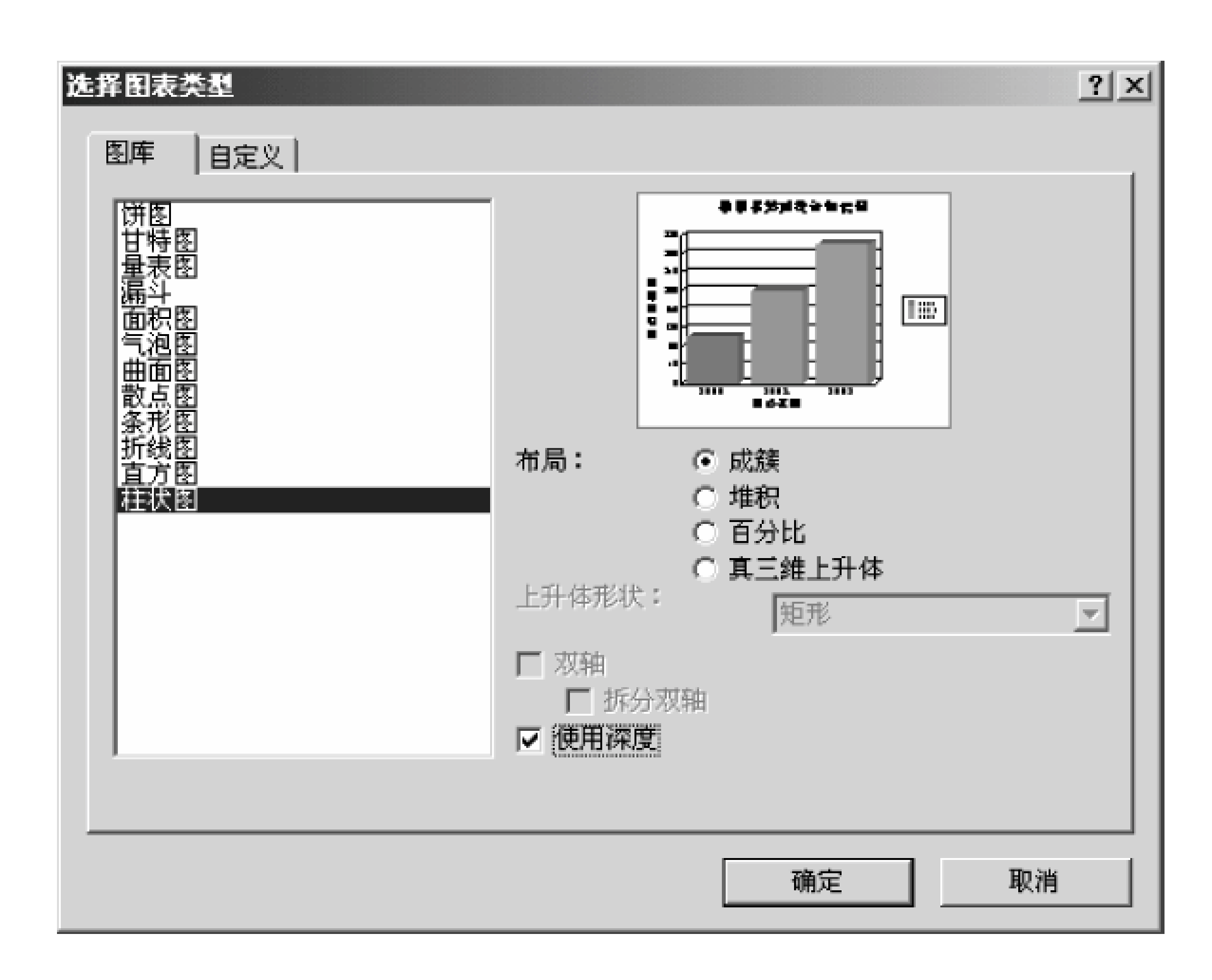


图 9-40　“选择图表类型”对话框

```
protected void Page_Load(object sender, EventArgs e)
{
        SaleDataSet saleDataSet = new SaleDataSet();
SaleDataSetTableAdapters.vw_OrderSaleDetailTableAdapter saleDa =
new SaleDataSetTableAdapters.vw_OrderSaleDetailTableAdapter();

        saleDa.Fill(saleDataSet.vw_OrderSaleDetail);
            CrystalReportSource1. ReportDocument. Load ( Server. MapPath ( " ～/Report/
CrystalReportChart.rpt"));
        CrystalReportSource1.ReportDocument.SetDataSource(saleDataSet);
}
```

程序运行后的结果如图 9-41 所示。

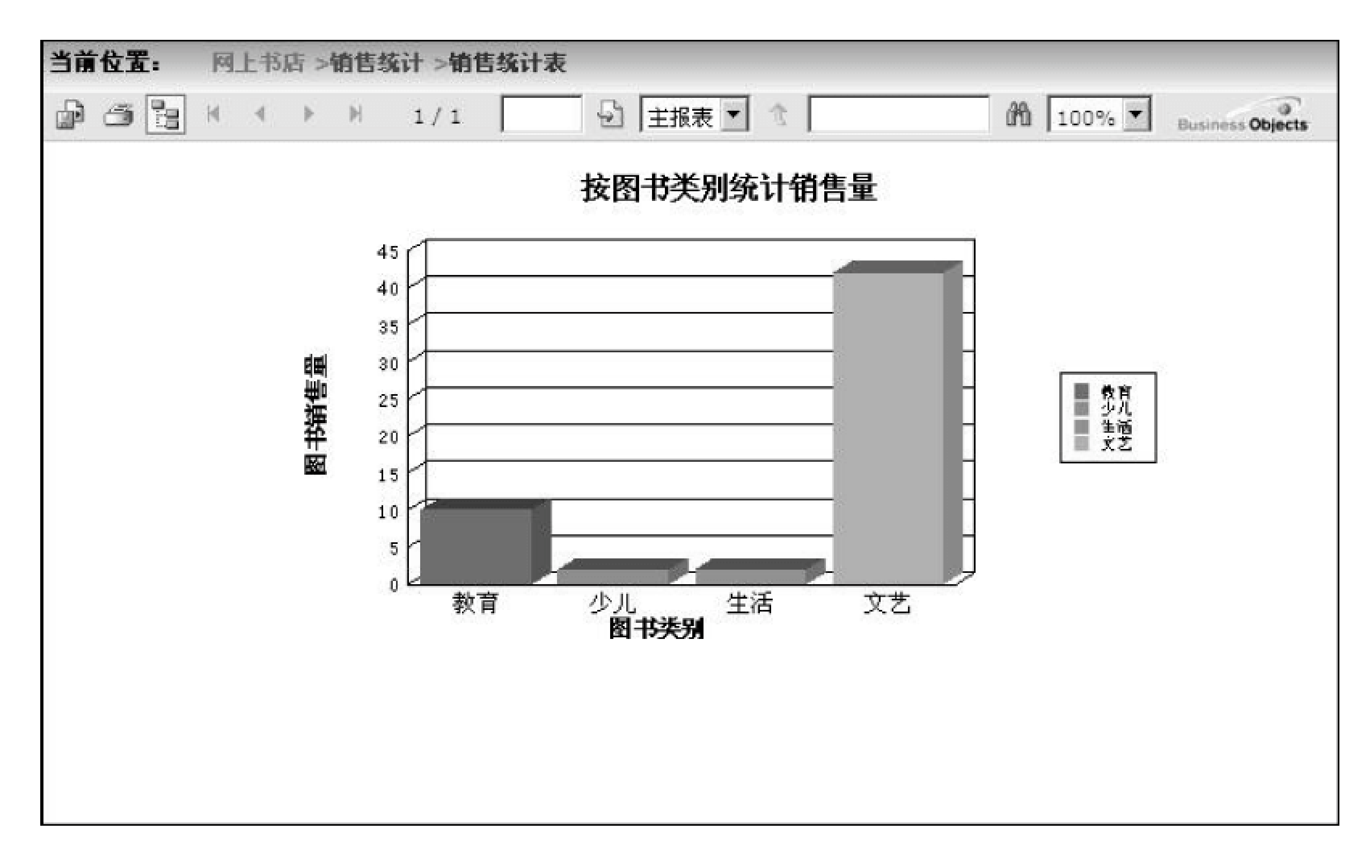


图 9-41　图书分类汇总销售量统计图

3. 任务完成总结

本任务中介绍了使用水晶报表的图表专家设计图书分类销售量汇总统计图的过程。整个设计过程主要分3个步骤进行，先设计订单销售明细数据集，并添加数据集到报表设计器，为报表提供数据支持；然后在报表文件中插入直方图，用图形方式描述图书分类统计销售量；最后将报表文件关联到页面的报表查看器控件上，通过报表页面文件呈现查询结果。

4. 课堂训练与知识拓展

参照任务9-4建立图形化报表的过程，使用饼形图呈现图书分类汇总销售量。

9.5 项目总结

项目9讨论了网上书店报表设计，使用VS 2005自带的水晶报表控件可以快速构建业务所需的报表。水晶报表使用报表专家功能来布局，水晶报表文件中的数据可以从与报表关联的数据集获取，水晶报表文件获取数据时有两种方法，使用Pull模式从数据库拉数据或使用Push模式向报表文件推送数据，在设计销售订单清单报表时分别使用Pull模式和Push模式实现。具有主从关系的数据一般用主从报表显示，水晶报表可以轻松设计主从报表，在设计订单基础信息和明细信息报表时使用了主从报表。还有一类报表是图表，图表可以直观反映数据之间的关系，设计网上书店分类销售量统计表时使用直方图表示。

9.6 项目实训

1. 任务描述

图书入库后需要打印图书入库明细清单、统计每个类目图书入库量。使用水晶报表功能设计图书入库明细清单和入库数量统计清单。

2. 任务要求

① 分别使用Pull模式和Push模式设计图书入库清单。
② 使用直方图或饼形图设计图书入库量分类统计报表。

项目10

网上书店Web认证与授权管理

10.1 项目介绍

项目 6 讨论了使用网站配置管理工具进行网站安全设置，并讨论了使用登录系列控件解决网上书店读者注册管理功能。本章将介绍网上书店 Web 认证和授权管理。几乎每个 Web 应用都会涉及针对不同的用户进行权限分配的工作，用户登录后根据不同的权限访问系统不同资源和实现不同的功能。在 ASP.NET 2.0 之前，开发人员需要自己设计和开发这样的功能，ASP.NET 2.0 中推出了成员(Membership)服务和相关控件，大大减轻了完成该任务的工作量。本项目中将使用 Web 授权机制、成员资格管理和角色管理实现网上书店 Web 认证和授权管理。

10.2 项目分析

网上书店的 Web 认证和授权管理关键点需要解决两个主要对象的管理：成员资格管理和角色管理。成员资格管理主要是管理具体用户，包括创建用户、删除用户、修改用户密码和用户授权等。成员资格描述的是用户的身份，而角色则是用户所能做的事情。比如，网上书店管理有若干事情要做，并不是一个人将所有事情做完，也不是每个人都需要做每件事情，而是需要对事情进行分类，如有处理订单的事情、有维护系统的事情，有管理财务的事情，当事情进行分类后才安排相应的人去做这些事情，我们把所有类别的事情可以定义为角色。因此，用户、角色和权限的关系可以是这样的一种关系：用户可以进入到角色中也可以从角色中离开，权限是赋予角色的，归属于角色的用户继承角色的权限。

本项目中基于上述思想来讨论网上书店的 Web 认证与授权管理。完成项目需要分 3 个任务进行。第一个任务使用 Membership 类和 MembershipUser 类创建用户管理系统，实现用户注册、用户登录和修改用户密码的功能，这一任务主要是进行成员资格管理。第二个任务使用 ASP.NET 网站管理工具对网上书店站点进行角色配置和授权，介绍 ASP.NET 网站管理工具在 Web 认证和授权上的应用，使用 ASP.NET 2.0 网站管理工具创建网上书店的用户角色，并建立访问规则，实现用户管理。第三个任务使用 Roles 类创建和管理角色，创建一个简单的网上书店角色管理系统，实现添加或删除角色，实现将用户分配到角色和从角色中删除用户等一些基本功能。

10.3 相关知识

Web站点中创建页面供用户浏览访问。这些页面可以分成两种类型：一类是无须用户提供凭证，可以允许所有用户直接访问的页面；另一类是只允许部分用户访问，这部分用户必须向Web站点提交用户凭证才能够访问受限资源。提交用户凭证涉及身份验证和授权。所谓的身份验收是指确定请求用户身份的行为，就是告诉服务器发出请求的用户是谁。当用户提交凭证通过身份认证后需确定此用户是否可以访问特定资源，这个过程称为授权。

1. Web应用认证

Web认证是一个过程，用户可以通过这个过程验证他们的身份，即解决在应用中“我是谁?”的问题，应用程序通过系统验证后的标识可以定位到唯一的用户。通常用户根据已有凭证进入登录页面，如已输入被确认的用户名或密码。ASP.NET 2.0提供了4种验证方式：Windows验证、Passport验证、None验证和Forms验证，默认为Windows验证。

（1）Windows验证

Windows验证适合在企业内部Intranet站点中使用，该验证方式基于Windows用户账号和用户组，可有效控制用户对未授权信息访问。如企业Intranet为每个雇员分配了Windows账号，这些账号可能包含在不同用户组中。公司的一些重要文档只允许中层主管访问，其他的人员不能访问，可以使用Windows验证来防止其他组用户访问，从而起到保护文档的作用。

Windows验证依靠IIS执行所需的身份验证。IIS提供了多种验证机制来验证用户身份，包括匿名身份验证、集成Windows身份验证、基本身份验证和Windows域服务器摘要式身份验证等。在ASP.NET中使用WindowsAuthenticationModule类实现Windows验证，WindowsAuthenticationModule类负责创建WindowsPrincipal和WindowsIdentity对象来表示经过身份验证的用户，并且负责将这些对象附加到当前的Web请求。在IIS中配置Windows认证的步骤如下：

① 打开IIS。在“控制面板”中打开“管理工具”，然后选中“Internet信息服务”，双击打开。

② 打开身份验证对话框。右键选中默认网站下的任一虚拟目录，选择属性，打开“虚拟目录属性”对话框。在此对话框中选择“目录安全性”选项卡，在匿名访问和身份验证控制选项里单击“编辑”，打开“身份验证”对话框。

③ 配置Windows身份验证。在身份验证对话框中勾选中“集成Widows验证”复选框。单击“确定”按钮。在IIS中配置Windows认证完成。

（2）Passport验证

Passport验证是一个用户身份验证服务。Passport提供了一个可以在多个网站共享的验证机制，可以让用户根据自己的电子邮件地址和密码，在任何支持Passport验证服务的Web站点上访问资源，并且在注销离开时，所有Passport相关信息都会清除。因此，

Passport 验证适合跨站应用。启用 Passport 验证的 Web 站点依靠 Passport 中央服务器来验证用户身份，而不是依靠它们自己的专用身份验证系统。Passport 中央服务器并不授权或拒绝特定用户访问单个启用 Passport 的站点，而是由网站来控制用户的权限。

（3）None 验证

在 None 验证方式下 ASP.NET 将不对请求进行任何附加验证。None 验证方式适用两种情况。一种是不希望对用户进行验证，例如，网上书店的图书展示页面，可以供任何用户访问。另一种是 ASP.NET 提供的身份验证方式不满足应用程序对身份验证的需要，需要实现自定义身份验证机制。

（4）Forms 验证

Forms 验证是多数 Web 应用程序使用的验证方式。Forms 验证本身并不能进行验证，只是使用自定义的用户界面收集用户信息，最终通过自定义代码实现验证。

Web 应用认证的设置方式，将 Web. Config 文件的<authentication>配置节的 mode 属性设置 Windows、Forms、Passport 或 None 中的一项即可。

```
<configuration>
    <system.web>
        <authentication mode = "[Windows|Forms|Passport|None]"/>
    </system.web>
</configuration>
```

2. 用户授权

用户授权是指确定经身份验证的用户是否可以访问请求资源的过程。在 ASP.NET 2.0 中主要包括以下授权项。

（1）文件授权

文件授权由 FileAuthorizationModule 类来执行。它检查 .aspx 或 .asmx 处理程序文件的访问控制列表（ACL）以确定用户是否应该具有对文件的访问权限。ACL 权限用于验证用户的 Windows 标识（如果已启用 Windows 身份验证）或 ASP.NET 进程的 Windows 标识。

（2）URL 授权

URL 授权由 UrlAuthorizationModule 执行，它将用户和角色映射到 ASP.NET 应用程序中的 URL。这个模块可用于有选择地允许或拒绝特定用户或角色对应用程序的访问权限。

在实际开发过程中 URL 授权应用最为广泛，首先为用户设置角色，然后再对角色进行 URL 授权，这样提高了授权管理的效率和可控性。下面重点介绍 URL 授权。

使用 URL 授权方式主要处理 3 个问题。一是允许或者拒绝特定用户或用户组访问；二是允许或拒绝特定角色访问；三是允许或拒绝基于不同 HTTP 提交方式的访问，Get 方式或 Post 方式。必须对 Web. Config 文件的<authorization>配置节进行设置。在<authorization>配置节中包含两个重要的子配置节<allow>和<deny>。

```
<authorization>
  <[allow|deny] users roles verbs />
```

```
</authorization>
```

allow 或 deny 元素是必需的。必须指定 users 或 roles 属性。可以同时包含二者，但这不是必需的。verbs 属性可选。allow 和 deny 元素分别授予访问权限和撤销访问权限。每个元素都支持表 10-1 所示的属性。

表 10-1 URL 授权元素属性

属性	说　明
users	标识允许或拒绝访问资源的用户。允许使用逗号分隔的列表来引用多个用户。此外，可以用问号（?）标识匿名用户；可以用星号（*）指定所有经过身份验证的用户
roles	标识允许或拒绝访问资源的角色。允许使用逗号分隔的列表来引用多个角色
verbs	定义操作 HTTP 提交方式，如 GET、HEAD 和 POST。默认值为“*”，标识支持所有的 HTTP 提交

下面的示例对 Kim 标识和 Admins 角色的成员授予访问权限，对 John 标识和所有匿名用户拒绝访问权限：

```
<authorization>
  <allow users="Kim"/>
  <allow roles="Admins"/>
  <deny users="John"/>
  <deny users="?"/>
</authorization>
```

下面的 <authorization>节演示如何授予 John 标识的访问权限并拒绝所有其他用户的访问权限：

```
<authorization>
  <allow users="John"/>
  <deny users="*"/>
</authorization>
```

可以使用逗号分隔的列表为 users 和 roles 属性指定多个实体，如下面的示例所示：

```
<allow users="John, Kim, contoso\Jane"/>
```

下面的示例允许所有用户对某个资源执行 HTTP GET 操作，但是只允许 Kim 标识执行 POST 操作：

```
<authorization>
  <allow verbs="GET" users="*"/>
  <allow verbs="POST" users="Kim"/>
  <deny verbs="POST" users="*"/>
</authorization>
```

在实现用户授权过程中，应遵循以下两个应用规则：一是应用程序级别的配置文件中包含的规则优先级高于继承的规则。系统通过构造一个 URL 的所有规则的合并列表，其中最近(层次结构中距离最近)的规则位于列表头，来确定哪条规则优先。二是给定应用程序一组合并的规则，ASP.NET 从列表头开始，检查规则直至找到第一个匹配项为止。

ASP.NET 的默认配置包含向所有用户授权的 ＜allow users＝"＊"＞ 元素。(默认情况下,最后应用该规则。)如果其他授权规则都不匹配,则允许该请求。如果找到匹配项并且它是 deny 元素,则向该请求返回 401 HTTP 状态代码。如果 allow 元素匹配,则模块允许进一步处理该请求。

3. 成员资格管理

ASP.NET 成员资格可以验证和管理 Web 应用程序的用户信息。它提供验证用户凭据、创建和修改成员资格用户及管理用户设置(如密码和电子邮件地址)等功能。ASP.NET 成员资格主要用于 ASP.NET Forms 身份验证。

ASP.NET 成员资格可以将用户信息保存在所选数据源中,数据的操作类主要封装在成员资格管理 API 和成员资格提供程序中,因此,只需很少的代码就能实现成员资格管理,提高了应用程序开发效率。

ASP.NET 成员资格主要由内置成员资格提供程序组成,这些程序与数据源及 membership 静态类进行通信。从 ASP.NET 代码调用 membership 类以执行用户验证和管理。

(1) Membership 类

在 ASP.NET 应用程序中,Membership 类用于验证用户凭据并管理用户设置(如密码和电子邮件地址)。Membership 类可以独自使用,或者与 FormsAuthentication 一起使用以创建一个完整的 Web 应用程序或网站的用户身份验证系统。

Membership 类提供的功能可用于以下方面:

① 创建新用户。

② 将成员资格信息(用户名、密码、电子邮件地址及支持数据)存储在 Microsoft SQL Server 或其他类似的数据存储区。

③ 对访问网站的用户进行身份验证。可以以编程方式对用户进行身份验证,也可以使用 Login 控件创建一个只需很少代码或无须代码的完整的身份验证系统。

④ 管理密码,包括创建、更改、检索和重置密码等。可以选择配置 ASP.NET 成员资格以要求一个密码提示问题及其答案来对忘记密码的用户的密码重置和检索请求进行身份验证。

Membership 类的 CreateUser 方法将新用户添加到数据存储区,并返回新创建用户的 MembershipUser 对象。CreateUser 方法的最后一个输出参数 status 用于指示用户创建成功或失败的原因。参数 status 的数据类型为 MembershipCreateStatus 类型。MembershipCreateStatus 枚举指示创建新用户的尝试的成功或失败。如果 CreateUser 操作失败,则 MembershipCreateStatus 枚举描述失败的原因。CreateUser 的语法如下:

```
public static MembershipUser CreateUser {
    string username,                //用户名
    string password,                //密码
    string email,                   //电子邮箱
    string passwordQuestion,        //密码问题
    string passwordAnswer,          //问题答案
    bool  isApproved,               //是否批准新用户登录
```

```
        Object providerUserKey,            //用户标识
        out MembershipCreateStatus status  //指示用户成功创建或创建失败的原因
}
```

MembershipCreateStatus 枚举值可以参考表 10-2。

表 10-2 MembershipCreateStatus 返回类型枚举值

成员名称	说明
Success	创建用户成功
InvalidUserName	在数据库中未找到用户名
InvalidPassword	密码的格式设置不正确
InvalidQuestion	密码提示问题的格式设置不正确
InvalidAnswer	密码提示问题答案的格式设置不正确
InvalidEmail	电子邮件地址的格式设置不正确
DuplicateUserName	用户名已存在于应用程序的数据库中
DuplicateEmail	电子邮件地址已存在于应用程序的数据库中
UserRejected	因为提供程序定义的某个原因而未创建用户
InvalidProviderUserKey	提供程序用户键值的类型或格式无效
DuplicateProviderUserKey	提供程序用户键值已存在于应用程序的数据库中
ProviderError	提供程序返回一个未由其他 MembershipCreateStatus 枚举值描述的错误

Membership 类依赖于成员资格提供程序与数据源通信。.NET Framework 包括一个 SqlMembershipProvider（将用户信息存储在 Microsoft SQL Server 数据库中）和一个 ActiveDirectoryMembershipProvider（允许在 Active Directory 或 Active Directory 应用程序模式（ADAM）服务器上存储用户信息）。还可以实现一个自定义成员资格提供程序与可由 Membership 类使用的其他类似的数据源进行通信。自定义成员资格提供程序将继承 MembershipProvider 抽象类。

默认情况下，ASP.NET 成员资格可支持所有 ASP.NET 应用程序。默认成员资格提供程序为 SqlMembershipProvider 并在计算机配置中以名称 AspNetSqlProvider 指定。SqlMembershipProvider 的默认实例配置为连接到 Microsoft SQL Server 的一个本地实例。

（2）MembershipUser 类

MembershipUser 类主要实现对具体用户的信息检索、密码管理等功能。Membership 类与 MembershipUser 类之间有着密切关系，体现在两个方面：一是二者都是成员资格管理 API 的核心类，都是实现与成员资格管理有关的功能；二是利用 Membership 类中都是有关创建用户、获取用户等方法，能够创建 MembershipUser 类对象，由此，可以对具体用户的信息进行检索与管理。

4. 角色管理

网上书店系统功能对不同用户进行授权访问。授权访问可以通过角色来管理，把用户映射到角色上，然后再给角色授权，映射到角色的用户享有角色内的权限。ASP.NET 2.0 提供的基于角色的授权方式将整个控制过程分为两个步骤：首先，访问权限与角色关联；其次，角色与用户关联，从而实现了用户与访问权限的逻辑分离。管理人员授权时仅为角色

授权,其影响的是角色中多个用户,这种设计方式提高了权限管理的效率与可控性。当用户的职位发生变化时,只要删除为该用户分配的角色,然后设置代表新职位的角色即可。委派用户到角色不需要复杂的技术,可由行政管理人员执行,配置权限到角色的工作比较复杂,需要一定的技术,可以由专门的技术人员来承担,这一过程与现实中的工作过程一致。

建立角色的主要目的是提供一种管理用户组访问规则的便捷方法。创建用户,然后将用户分配到角色。典型的应用是创建一组受限访问的页面,这组页面只有某些用户可以访问。通常的做法是将这些受限制的页面单独放在一个文件夹内。然后,可以使用网站管理工具定义允许和拒绝访问受限文件夹的规则。例如,可以配置站点以使成员和经理可以访问受限文件夹中的页面,并拒绝其他所有用户的访问。如果未被授权的用户尝试查看受限制的页面,该用户会看到错误消息或被重定向到指定的错误处理页面。

若要使用角色,必须能够识别应用程序中的用户,以便可以确定用户是否属于特定角色,可以对应用程序进行配置,以两种方式建立用户标识:Windows 身份验证和 Forms 身份验证。如果应用程序在局域网(即在基于域的 Intranet 应用程序)中运行,则可以使用用户的 Windows 域账户名来标识用户。在这种情况下,用户的角色是该用户所属的 Windows 组。在 Internet 应用程序和其他不适合使用 Windows 账户的方案中,可以使用 Forms 身份验证来建立用户标识。对于此任务,通常是创建一个页面,用户可以在该页面中输入用户名和密码,然后对用户凭据进行验证。ASP.NET 登录控件可以执行其中的大部分工作,也可以创建登录页面并使用 FormsAuthentication 类建立用户标识。

角色管理还提供了一个 API,可使用该 API 以编程方式确定用户是否属于某角色。这样能够编写利用角色的代码,基于用户是谁和基于用户所属的角色来执行所有应用程序任务。

若要使用 ASP.NET 角色管理,需要在应用程序的 Web.Config 文件中启用它:

```
<roleManager
    enabled="true"
    cacheRolesInCookie="true" >
</roleManager>
```

10.4　项目实施

下面通过 3 个任务来讨论网上书店认证与授权管理。

10.4.1　任务 10-1　使用 Memship 类和 MemshipUser 类创建用户管理系统

1. 任务介绍

项目 6 中介绍了使用登录系列控件完成网上书店用户登录与注册管理功能。当创建一个多用户站点时,对用户进行管理也可以使用 Membership 类和 MembershipUser 类来实现,本任务中将介绍使用 Membership 类和 MembershipUser 类创建网上书店的用户管理系统,此系统主要实现用户注册,用户登录,更改密码功能。

2. 任务分析

一个简单的网站用户管理系统至少包含三个功能：用户注册、用户登录、修改密码。本任务中将首先使用 Membership 类和 MembershipCreateStatus 类实现用户注册功能，调用 Membership 类的 CreateUser 方法，将用户输入的注册信息写入数据库，对数据库读写操作是否成功，系统会返回一个 MembershipCreateStatus 枚举类型的状态码，通过状态码可以知道成功与否，以及不成功的原因。在实现用户的登录功能时使用 Membership 类 ValidateUser 方法对输入的用户名和密码进行验证，如果验证通过就可以调用 FormsAuthentication 类的 SetAuthCookie 方法将用户名写入 Cookie，并创建身份验证票，以供在授权页面使用。修改密码需要授权，用户登录后进入修改密码页面，首先判断 User. Identity. IsAuthenticated 是否为真，即判断用户是否经过了身份验证。如果是授权用户，则调用 Membership 类的 GetUser 方法获取用户对象，然后调用用户对象的 ChangePassword 方法完成用户密码修改。

(1) 实现用户注册功能

在网上书店项目解决方案中添加一个 Users 文件夹，然后在 Users 文件下添加文件 AddUser. aspx，打开设计界面，设计用户输入信息：用户名、输入密码、重新输入密码、邮箱、密码问题、密码答案。对每个文本输入框使用必填验证控件进行输入必填验证，对输入密码和重新输入密码使用比较验证控件进行一致性校验，保证两次输入密码一致。设计界面如图 10-1 所示。

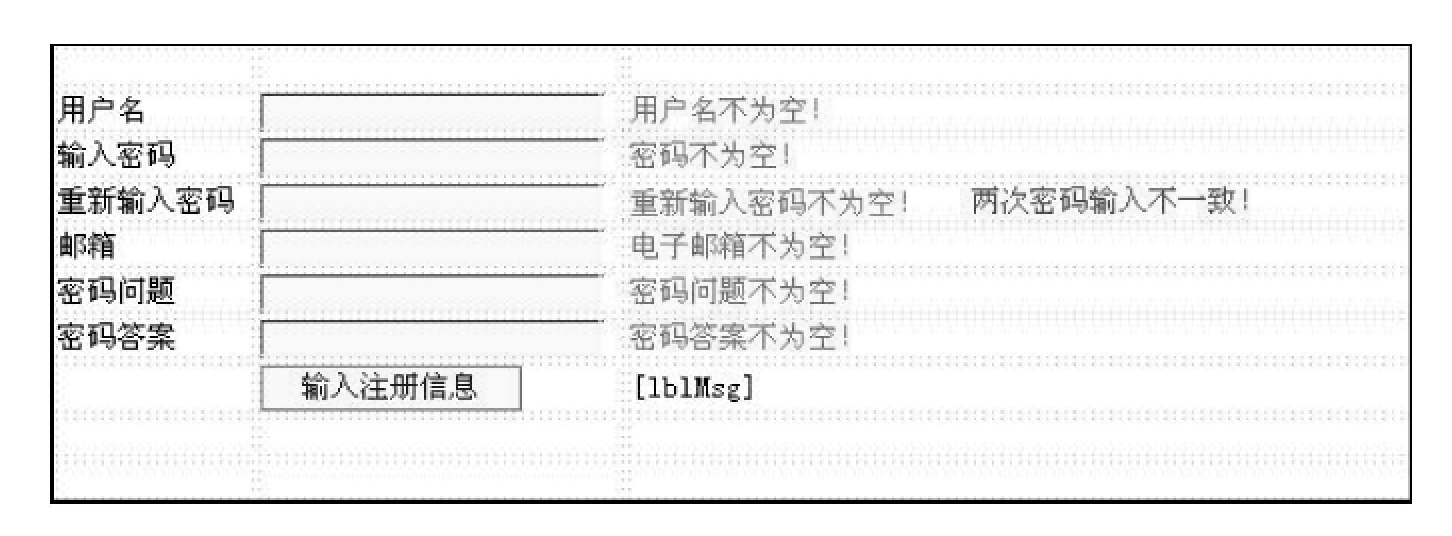


图 10-1　注册输入界面设计

在页面的“输入注册信息”按钮的 Click 事件中输入验证添加用户是否成功的代码：

```
protected void Button1_Click(object sender, EventArgs e) {
        MembershipCreateStatus status;
        Membership.CreateUser(txtUserName.Text
                             , txtPassword.Text
                             , txtEmail.Text
                             , txtQuestion.Text
                             , txtAnwser.Text
                             , true
                             , out status);

        if (status == MembershipCreateStatus.Success){
            lblMsg.Text = "恭喜!注册成功!";
        }
```

```
        else{
            lblMsg.Text = "注册失败!请检查!";
        }
}
```

代码中使用 Membership 类的 CreateUser 方法注册用户，如果用户注册成功，则输出参数 status 返回值是表 10-2 中的“Success”。

浏览 AddUser.aspx 页面，创建一个新用户，用户详细信息如下：

```
用户名: zhangsan
密码: zhangsan@126.com
邮箱: zhangsan@163.com
密码问题: 我是谁?
密码答案: zhangsan
```

单击“添加用户”按钮，页面显示添加用户成功的信息“恭喜！注册成功！”。打开 ASP.NET 网站管理工具，我们看到刚才添加的用户已经存在。

(2) 实现用户登录功能

在 Users 文件夹下添加文件 Login.aspx，设计用户登录界面，登录界面中的输入内容为用户名和密码，分别用文本控件接受输入。在界面上再放置两个 Button 按钮，一个 Text 属性设置为“登录”；另一个 Text 属性设置为“添加用户”，如图 10-2 所示。“添加用户”按钮的 PostBackUrl 属性指向 AddUser.aspx 文件，单击该按钮时，可以导向注册界面。

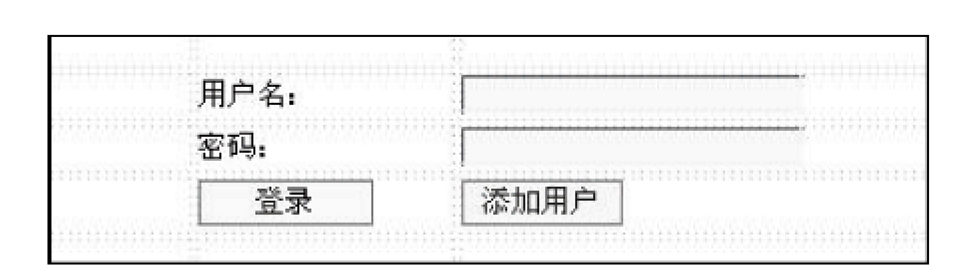


图 10-2　用户登录

编写“登录”按钮的 btnLogin _Click 事件：

```
protected void btnLogin_Click(object sender, EventArgs e) {
        if (Membership.ValidateUser(txtUserName.Text, txtPassword.Text)) {
            FormsAuthentication.SetAuthCookie(txtUserName.Text, false);
            Response.Redirect("~/Web/MyBookShop.aspx");
        }
}
```

代码中使用 Membership 类的 ValidateUser 方法验证用户，需要两个参数“用户名”和“密码”。如果验证通过方法返回布尔值 True，否则返回布尔值 False。验证通过后使用 FormsAuthentication 类的 SetAuthCookie 方法将用户名写入 Cookie，自动创建身份验证票。方法 SetAuthCookie 的第二个参数是个逻辑值，值为 True，表示浏览器关闭后，验证票仍然有效；值为 False，则关闭浏览器后验证票失效，需要重新登录。

上述代码验证需要配置 Web.Config 文件：

```
<authentication mode = "Forms">
    <forms loginUrl = "~/Web/Users/Login.aspx"/>
</authentication>
```

在目标页面文件 MyBookShop. aspx 的页面隐藏类 MyBookShop. aspx . cs 的 Page_Load 方法中使用 User. Identity. IsAuthenticated 的返回值来判断用户是否有权访问该页。如果验证票有效，则返回值为 True，表示有权访问该页，否则导向登录页面强制用户登录。

```
if (!User.Identity.IsAuthenticated)
{
      Response.Redirect("~/Web/Login.aspx");
}
```

(3) 实现更改登录用户密码功能

在 Users 文件夹下添加页面文件 ChangePassword. aspx，设计修改密码界面，密码修改需要输入用户名和原密码，使用 Forms 认证，用户名从 User. Identity. Name 中获取，原密码由界面输入。需要修改的新密码从界面输入，新密码需要进行一次重复验证，可以由比较验证控件完成。页面上放置一个 Button 按钮，按钮显示“更改密码”。页面设计如图 10-3 所示。

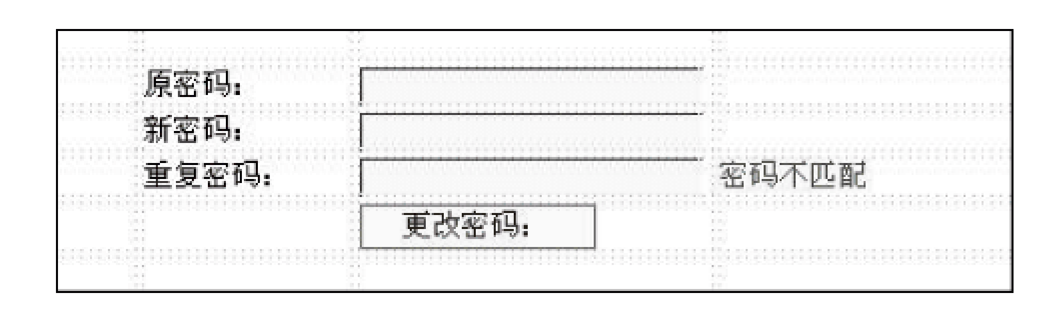


图 10-3 更改密码

“更改密码”按钮的 Click 事件代码如下：

```
protected void btnChangePassword_Click(object sender, EventArgs e) {
        MembershipUser user = Membership.GetUser(User.Identity.Name);
        if (user.ChangePassword(txtOldPassword.Text, txtNewPassword.Text)) {
            Response.Write("用户密码更改成功");
        }
}
```

使用 Membership 类的 GetUser 方法获取用户对象，参数为用户名，返回值数据类型为 MembershipUser 类型，将结果保存到 user 变量中，调用 user 的 ChangePassword 方法完成密码修改，ChangePassword 方法需要两个参数：第一个参数是旧密码；第二个参数是新密码。

3. 任务完成总结

本任务主要介绍使用 Membership 类和 MembershipUser 类实现编写代码完成网上书店用户管理系统。实现的过程分为 3 个步骤：用户注册、用户登录和密码修改。用户登录使用 Forms 身份认证，通过调用 FormsAuthentication 类的方法 SetAuthCookie 将用户名写入 Cookie，创建身份验证票，进入授权页面首先验证用户的合法身份，通过验证才可以访问授权页。修改密码是通过 ChangePassword 方法完成的，该方法封装在 MembershipUser 类中。

4. 课堂训练与知识拓展

一个比较完整的用户管理系统，还应包含修改用户注册信息、删除用户账户等功能。使

用代码完善网上书店的用户管理功能，增加修改用户注册信息的功能和删除用户注册账户的功能。

10.4.2 任务 10-2 使用 ASP.NET 网站管理工具对站点进行角色配置和授权

1. 任务介绍

ASP.NET 2.0 网站管理工具除了可以创建用户、管理用户和选择身份验证类型外，还可以启用角色，并创建和管理角色，将用户分配到角色，同时网站管理工具还可以创建访问规则和管理访问规则。本任务将使用 ASP.NET 2.0 网站管理工具创建网上书店的用户角色，并建立访问规则，实现用户管理。

2. 任务分析

打开 ASP.NET 网站管理工具界面的“安全”选项卡，应用程序设置第 2 列为角色设置，第 3 列为访问规则设置。使用角色设置需先启用角色，通过启用角色可以将“创建和管理角色”功能激活。使用激活的功能可以完成添加角色、删除角色和将用户分配到角色等功能。角色创建好后可以建立访问规则，访问规则作用于文件夹，在建立功能文件时，将功能相近的文件放在同一文件夹下，便于授权管理。使用规则允许或拒绝角色对文件夹的访问。

(1) 使用管理工具创建和管理角色

选择菜单栏功能菜单“网站”下的“ASP.NET 配置”子菜单，可以打开“ASP.NET 网站管理工具”功能页面。使用网站管理工具“安全”选项卡可以管理用户、角色和访问规则。如图 10-4 所示。

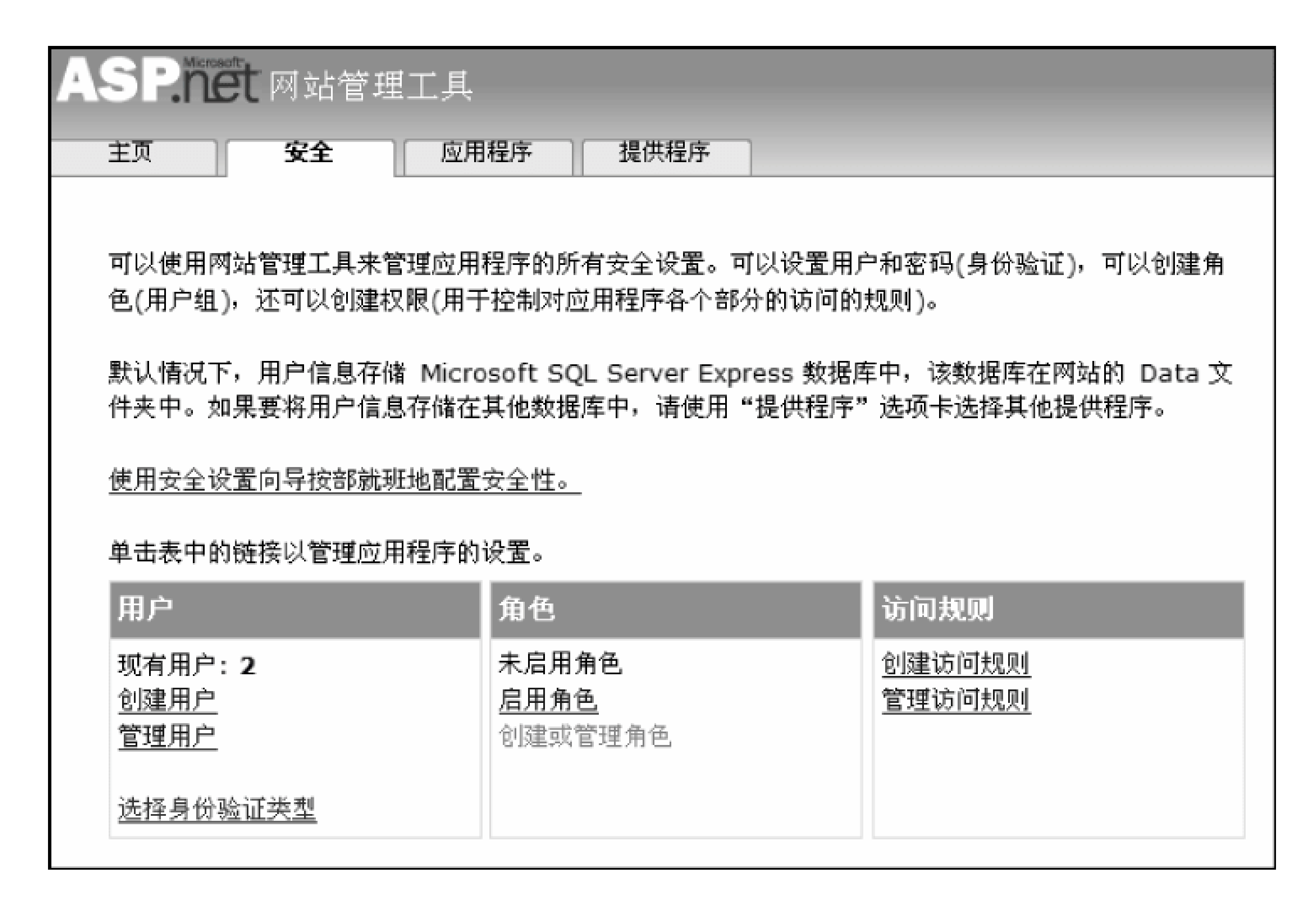


图 10-4　网站管理工具“安全”选项卡

在网站管理工具安全页面中，单击“启用角色”激活“创建或管理角色”功能。当启用角色管理后在应用程序根目录下的 Web.Config 的<system.web>节点下已经多了一项配置节<roleManager enabled="true" />。

修改<roleManager enabled="true" />配置项，使用<clear/>配置项清除原来的角色连接提供程序。重新定义角色连接程序 AspNetSqlRoleProvider，使其指向 BookShop 应用程序，使用 LocalSqlServer 连接提供程序访问 NetBook 数据库。代码如下：

```
<roleManager enabled="true">
    <providers>
      <clear/>
      <add name="AspNetSqlRoleProvider"
           connectionStringName="LocalSqlServer" applicationName="BookShop"
           type="System.Web.Security.SqlRoleProvider"/>
    </providers>
</roleManager>
```

使用“创建或管理角色”功能可以进入创建新角色界面，如图 10-5 所示。创建两个新角色“订单管理员”和“系统管理员”。

创建新角色

新角色名称：[ ] 添加角色

角色名称	添加/移除用户	
订单管理员	管理	删除
系统管理员	管理	删除

图 10-5　使用网站管理工具创建新角色

选择指定角色的管理功能可以为角色分配用户。单击“系统管理员”角色的“管理”功能打开为角色分配用户界面。“搜索依据”下拉列表框选择“用户名”，在搜索文本框中输入用户名，单击“查找用户”按钮查找指定用户，也可以使用“*”和“?”通配符批量查找用户。使用“*”通配符搜索用户的结果如图 10-6 所示。将用户“zhangsan”勾选到“系统管理员”角色中。使用同样的过程将用户“netbook”分配到角色“订单管理员”中。



图 10-6　分配用户到角色

(2) 使用管理工具建立和管理规则

访问规则管理用于用户或者角色对于路径的访问权限进行授权，在图 10-4 网站安全界面中单击“创建访问规则”链接，打开图 10-7 所示的“添加新访问规则”界面。在这里选中 MangOrder 文件夹，然后选择“角色”，选中先前创建的角色“订单管理员”，然后在权限部分

选择“允许”。这样一种组合就表示允许订单管理员这个角色访问 MangOrder 文件夹。要达到授权要求，除了允许订单管理员访问文件夹之外，还需要禁止其他人或角色访问 MangOrder 文件夹。在单击“确定”按钮保存这个访问规则之后，再次单击“创建访问规则”链接来创建一个拒绝规则：指定所有用户对于 MangOrder 文件夹的访问权限为拒绝，如图 10-8 所示，单击“确定”按钮完成规则设置。

图 10-7　使用网站管理工具添加允许访问规则

图 10-8　使用网站管理工具添加拒绝访问规则

返回到图 10-4 网站安全界面单击“管理访问规则”链接，打开“管理访问规则”界面如图 10-9 所示。单击 MangOrder 文件夹，将展示两条授权规则，订单管理员允许访问该文件夹，其他人或角色被拒绝访问该文件夹。



图 10-9　使用网站管理工具管理访问规则

经过以上配置过程，在 MangOrder 文件夹下将产生一个 Web.Config 文件，内容如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<configuration>
    <system.web>
        <authorization>
            <allow roles = "订单管理员" />
            <deny users = " * " />
```

```
        </authorization>
    </system.web>
</configuration>
```

配置节“<allow roles="订单管理员" />”允许订单管理员角色访问 MangOrder 文件夹，配置节“<deny users=" * " />”拒绝其他用户访问该文件夹。

(3) 测试角色和访问规则

打开 NetBook 应用程序网站，进入登录界面，分别使用“netbook”账户和“zhangsan”账户登录网站，单击“订单处理”功能链接，“netbook”账户属于“订单管理员”角色可以正常访问该页面，打开销售订单列表。“zhangsan”账户不能访问该页，被导向登录页面。

3. 任务完成总结

本任务中主要介绍了 ASP.NET 网站管理工具建立角色和管理角色的过程，以及创建规则和管理规则的过程。规则作用于角色，决定角色是否对文件夹具有访问权限。用户从属于角色，用户通过角色取得访问文件的权限。

4. 课堂训练与知识拓展

建立“我的网店”角色和“销售统计”角色，创建账户“lisi”和“wangwu”，账户“lisi”是普通客户，账户“wangwu”是网店的管理人员，将“lisi”分配到角色“我的网店”，将“wangwu”分配到角色“销售统计”。文件夹 UserShop 下保存跟用户相关的操作功能，如购物车、订单查询等，建立访问规则将文件夹 UserShop 指派给“我的网店”角色。文件夹 SaleSum 下保存销售统计功能，建立访问规则将该文件夹指派给“销售统计”角色。验证账户“lisi”和“wangwu”各自的访问权限。

10.4.3 任务 10-3 使用 Roles 类建立和管理角色

1. 任务介绍

本章任务 10-2 介绍了使用 ASP.NET 网站管理工具建立和管理角色，本任务中将介绍使用 Roles 类创建一个简单的网上书店角色管理系统，实现添加或删除角色，为角色分配用户和删除用户等一些基本功能。

2. 任务分析

Roles 类提供了一组丰富的方法，实现对角色的管理。GetAllRoles 方法可以获取所有的角色列表，RoleExists 方法可以判断角色是否已经存储在数据库中，CreateRole 方法可以创建角色，DeleteRole 方法从数据存储区删除角色。以上方法是对角色本身的管理，除此之外，还提供了管理角色中用户的方法。IsUserInRole 方法判断用户是否包含在指定角色中，AddUserToRole 方法将用户添加到角色中，RemoveUserFromRole 方法可以从角色中移除用户，GetUsersInRole 方法可以获取角色中用户列表。本任务将使用上述方法实现创建角色和管理角色的功能。

对角色的管理过程分两个步骤完成：第一步实现角色的添加和删除；第二步实现添加

用户到角色和从角色中删除用户。

（1）实现角色的添加和删除

添加角色和删除角色界面设计如图 10-10 所示。

实现角色管理功能

输入角色名称 [ ] 添加角色

序号	角色名称	删除
数据绑定	数据绑定	删除
数据绑定	数据绑定	删除
数据绑定	数据绑定	删除
数据绑定	数据绑定	删除
数据绑定	数据绑定	删除

[lblMsg]

图 10-10　添加和删除角色

在解决方案 Roles 文件夹下添加文件 MangRoles.aspx，在 MangRoles.aspx 设计界面上添加文本控件 txtRoleName 用于输入角色，当单击“添加角色”按钮时，可以将文本框中角色添加到数据角色表中。角色清单列表使用 GridView 控件实现，控件的 ID 定义为 gvRoles。GridView 控件共设计成三列：序号、角色名称和删除。这三列均使用模板技术实现。第一列模板中添加一列序号，使用代码绑定技术实现，模板项中绑定代码“<％＃ Container.DataItemIndex＋1 ％>”。第二列模板项中放控件文本框绑定“角色”。第三列模板项中放功能按钮，实现删除角色的功能。

界面设计代码如下：

```
<fieldset style = "width:500px">
        <legend>实现角色管理功能</legend>
        <br />
        <asp:Label ID = "Label1" Width = "100" runat = "server"
                Text = "输入角色名称"/>
        <asp:TextBox ID = "txtRoleName" Width = "200" runat = "server">
        <asp:Button ID = "btnAdd" Width = "100" runat = "server"
                Text = "添加角色" OnClick = "btnAdd_Click" />
        <br />
        <br />
        <asp:GridView ID = "gvRoles" Width = "460px" runat = "server">
            <Columns>
                <asp:TemplateField HeaderText = "序号">
                    <ItemTemplate>
                        <% #Container.DataItemIndex + 1 %>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText = "角色名称">
                    <ItemTemplate>
                        <asp:TextBox ID = "txtRole" Width = "300px" runat = "server"
                            Text = "<% # Container.DataItem.ToString() %>"/>
                    </ItemTemplate>
                </asp:TemplateField>
```

```
                        <asp:TemplateField HeaderText = "删除">
                            <ItemTemplate>
                                <asp:LinkButton ID = "lbtDel" Width = "50px" runat = "server" Text = "删除"
                                  OnClientClick = "return confirm('是否删除该行?');"
                                      onclick = "lbtDel_Click"></asp:LinkButton>
                            </ItemTemplate>
                        </asp:TemplateField>
                    </Columns>
                </asp:GridView>
                <br />
                <br />
                <asp:Label ID = "lblMsg" ForeColor = "Red" runat = "server" Text = ""/>
    </fieldset>
```

可以使用 Roles 类的 GetAllRoles 方法获取所有的角色列表，GridView 控件绑定显示角色数据源的代码如下：

```
gvRoles.DataSource = Roles.GetAllRoles();
gvRoles.DataBind();
```

添加角色过程在“添加角色”按钮的 btnAdd_Click 事件中完成，事件代码中首先要做非空验证，然后判断角色是否已经存在，如果角色不存在，则使用 Roles 类的 CreateRole 方法将角色添加到数据库。代码如下：

```
if (!string.IsNullOrEmpty(txtRoleName.Text)) {
      if (Roles.RoleExists(txtRoleName.Text)) {
            lblMsg.Text = "此角色已存在";
      }
      else {
            Roles.CreateRole(txtRoleName.Text);
            lblMsg.Text = "添加角色成功";
      }
}
```

删除角色的实现过程稍微有些复杂，这里给出完整的方法。“删除”按钮对象的类型是 LinkButton，按钮对象事件函数 lbtDel_Click 第一个参数 sender 指按钮对象本身，使用时需做类型转换。“删除”按钮对象在 GridView 对象的单元格中，单元格对象包含在 GridView 的数据行对象中，所以连续使用两次 lbtDel 对象的 Parent 方法便可以定位到按钮所在的数据行，找到的数据行对象也需要做类型转换。在数据行中使用 FindControl 方法查找呈现“角色”的文本框对象 txtRole。从文本框对象中获取角色，调用 Roles 类的 DeleteRole 方法删除角色。进行删除操作后要重新进行绑定呈现角色列表操作。

删除角色的代码如下：

```
protected void lbtDel_Click(object sender, EventArgs e) {
        GridViewRow gvRow = ((sender as LinkButton).Parent.Parent as GridViewRow);
        TextBox txtRole = gvRow.FindControl("txtRole") as TextBox;

        if (Roles.RoleExists(txtRole.Text)) {
            if (Roles.DeleteRole(txtRole.Text)) {
```

```
                lblMsg.Text = "删除角色成功";
            }
        }

        gvRoles.DataSource = Roles.GetAllRoles();
        gvRoles.DataBind();
    }
```

(2) 实现从角色中移除用户或把用户添加到角色

在 Roles 文件夹下添加文件 AddUserToRole.aspx，使用“<asp:DropDownList ID="ddlRole" runat="server" />”呈现所有角色，下拉控件数据源通过 Roles 类的 GetAllRoles 方法获得。代码如下：

```
ddlRole.DataSource = Roles.GetAllRoles();
ddlRole.DataBind();
```

在设计界面上添加 GridView 控件，ID 定义为 gvRoleUsers，该控件用于显示 ddlRole 下拉控件中选中的角色内的用户。控件共三列，第一列为“序号”、第二列为“用户名”、第三列为“移除用户”功能按钮，实现从角色中移除用户。代码如下：

```
<asp:GridView ID="gvRoleUsers" Width="460px" runat="server">
    <Columns>
        <asp:TemplateField HeaderText="序号">
            <ItemTemplate>
                    <%#Container.DataItemIndex+1 %>
                </ItemTemplate>
          </asp:TemplateField>
          <asp:TemplateField HeaderText="用户名称">
                <ItemTemplate>
                    <asp:Label ID="lblUser" Width="300px" runat="server"
                            Text="<%# Container.DataItem.ToString() %>" />
                </ItemTemplate>
          </asp:TemplateField>
          <asp:TemplateField HeaderText="移除">
                <ItemTemplate>
                    <asp:LinkButton ID="lbtMoveUserFromRole"
                                Text="移除用户" runat="server"
                                onclick="lbtMoveUserFromRole_Click"/>
                </ItemTemplate>
          </asp:TemplateField>
     </Columns>
</asp:GridView>
```

使用 Roles 类的 IsUserInRole 方法判断用户是否包含在指定角色中，如果用户包含在角色中则使用 Roles 类的 RemoveUserFromRole 方法从角色中移除用户。使用 Roles 类的 GetUsersInRole 方法可以获取指定角色中的用户列表，结果绑定到 gvRoleUsers 控件上呈现。lbtMoveUserFromRole_Click 事件代码如下：

```
if (Roles.IsUserInRole(lblUser.Text, roleName)) {
     Roles.RemoveUserFromRole(lblUser.Text, roleName);
```

```
        gvRoleUsers.DataSource = Roles.GetUsersInRole(roleName);
        gvRoleUsers.DataBind();
    }
```

完成添加用户到角色的功能首先要显示所有用户，可以在 AddUserToRole.aspx 文件的设计界面上添加 GridView 控件，控件 ID 定义为 gvUsers，控件的结构与 gvRoleUsers 一致。控件的最后一列为“分配用户”功能按钮。按钮设计代码如下：

```
<asp:LinkButton ID="lbtAddUserToRole" Text="分配用户" runat="server"
                onclick="lbtAddUserToRole_Click"/>
```

可以使用 Membership 类的 GetAllUsers 方法获得所有用户的列表绑定到 gvUsers 控件上显示。代码如下：

```
gvUsers.DataSource = Membership.GetAllUsers();
gvUsers.DataBind();
```

在“分配用户”按钮的 lbtAddUserToRole_Click 事件函数中完成将用户分配到角色的过程。如果角色中没有要分配的用户，则使用 Roles 类的 AddUserToRole 方法将用户添加到指定角色中。代码如下：

```
if (!Roles.IsUserInRole(lblUser.Text,roleName)){
        Roles.AddUserToRole(lblUser.Text, roleName);
        gvRoleUsers.DataSource = Roles.GetUsersInRole(roleName);
        gvRoleUsers.DataBind();
}
```

3. 任务完成总结

本任务中主要介绍了使用 Roles 类完成添加角色、删除角色的过程，以及将用户分配到角色和从角色中移除用户的过程。在添加和删除角色时使用了 Roles 类的 RoleExists 方法判断角色是否存在，使用 CreateRole 方法和 DeleteRole 方法完成角色的添加和删除功能。在向角色中添加用户时，使用 Roles 类的 AddUserToRole 方法；而从角色中移除用户时，使用的是 Roles 类的 RemoveUserFromRole 方法。

4. 课堂训练与知识拓展

依据本任务中的步骤设计一个简单的角色管理程序，实现添加角色、删除角色、添加用户到角色和从角色中移除用户的功能，并实现不同角色的用户授权。

10.5 项目总结

本项目通过 3 个递进的任务介绍了网上书店 Web 认证和授权的实现过程。第一个任务讨论了使用成员资格管理技术设计网上书店用户管理系统，重点掌握成员资格管理类 Membership 类和 MembershipUser 类的用法。第二个任务重点分析使用 ASP.NET 网站管理工具进行角色配置和授权管理，需重点掌握 Web.Config 文件关于角色配置项设置以

及掌握 ASP.NET 网站管理工具的使用。第三个任务讨论了使用 Roles 类管理角色，以及角色和用户的关系，重点掌握 Roles 类的使用。

10.6　项目实训

1. 任务描述

建立一个基于 Web 方式的权限管理系统 Authorization，实现 Web 认证和授权。在 SQLServer 2005 中建立权限管理数据库，数据库名为 aspnetdb，使用 ASP.NET SQL Server 注册工具 Aspnet_regsql.exe 生成权限管理数据库的表结构。

2. 任务要求

① 打开 Web.config 文件，在＜system.web＞下添加＜authentication＞节点，并配置其参数，包括认证模式、登录页面等。

② 添加数据库连接（MembershipConnection），连到 aspnetdb。（注意：使用 SQL Server 2005 或 SQL Express 均可。）

③ 配置系统认证提供程序，使用默认提供程序 SqlMembershipProvider，数据库连接使用 MembershipConnection。

④ 配置系统授权提供程序，使用默认提供程序 SqlRoleProvider，数据库连接使用 MembershipConnection。

⑤ 使用 Memship 类和 MemshipUser 类完成新用户注册、登录和修改密码的操作。用户登录后使用 LoginName 控件显示当前登录用户名。

⑥ 在站点根目录新建 Members 子目录，并新增 Web.config 文件，通过配置 URL 授权，只允许 Members 角色的用户才可以访问该目录下的页面。

⑦ 在 Sales 目录下新建 Web.config 文件，通过配置 URL 授权，只允许 Sales 角色的用户才可以访问该目录下的页面。

⑧ 在 Members 目录下新建 MemberUpdate.aspx 页面，并添加两个 Label 和 TextBox 控件，用于显示用户名和用户 E-mail，只允许修改 E-mail，然后在 Page_Load 事件方法中通过成员资格 API 访问当前登录用户信息，并将信息输出到 TextBox 控件中。

⑨ 再添加一个“修改”按钮，并在按钮的单击事件中通过成员资格 API 修改用户的 E-mail。

参 考 文 献

[1] 微软公司. ASP.NET 2.0 Web 应用开发[M]. 北京：高等教育出版社，2007.

[2] 微软公司. .NET Framework 2.0 程序设计[M]. 北京：高等教育出版社，2007.

[3] 陈冠军. 精通 ASP.NET 2.0 典型模块设计与实现[M]. 北京：人民邮电出版社，2007.

[4] 郭靖. ASP.NET 开发技术大全[M]. 北京：清华大学出版社，2009.

[5] 刘亮亮，潘中强. 精通 ASP.NET 2.0 数据绑定技术 [M]. 北京：人民邮电出版社，2008.

[6] 张树亮，李超. ASP.NET 2.0+SQL Server 网络应用系统开发案例精解[M]. 北京：清华大学出版社，2006.

[7] 郝刚. ASP.NET 2.0 开发指南[M]. 北京：人民邮电出版社，2006.

[8] 刘基诚，李愈胜，刘卫卫译. Programming C# 中文版. 4 版[M]. 北京：电子工业出版社，2007.

[9] 朱晔. ASP.NET 第一步——基于 C# 和 ASP.NET 2.0[M]. 北京：清华大学出版社，2007.

[10] 陈冠军. 精通 ASP.NET 2.0 企业级项目开发[M]. 北京：人民邮电出版社，2007.

[11] (美)弗里曼等. Head First 设计模式(中文版)[M]. O'Reilly Taiwan 公司译. 北京：中国电力出版社，2007.

[12] 微软 msdn 官方网站. http://msdn.microsoft.com/zh-cn/asp.net/default.aspx.

[13] 微软 Pet Shop 4. http://msdn.microsoft.com/zh-cn/library/aa479070.aspx.